DOCTOR OF ENGINEERING (ENGD)

Improving the Pipeline for Stereo Post-Production

Willey, Stephen

Award date:
2017

Awarding institution:
University of Bath

Improving the Pipeline for Stereo Post-Production

submitted by

Stephen Alan Willey

for the degree of Doctor of Engineering

of the

University of Bath

Department of Computer Science

October 2017

Dedicated to my mum
Lynne Christina Willey
1961-2012

Contents

List of Figures

Acknowledgements

Thank you to my supervisory team: Phil Willis and Peter Hall of the University of Bath; Jian Zhang of Bournemouth University; and Jeff Clifford and Ted Waine of Double Negative. Thank you also to the examiners, Hassan Ugail and Darren Cosker. These people all helped to shape this research into a meaningful and coherent piece of work, rather than the hot mess it would otherwise have been.

I am grateful to the Centre for Digital Entertainment, Double Negative Ltd, and EPSRC for supporting this research. I would like to thank all members of the Double Negative Research and Development team, past and present, for all the guidance, discussion and feedback - particularly Mungo, who is the only person so far to have read this thesis in its entirety when they did not *have* to.

Thank you to all my family and friends for their support and encouragement over the years and, finally, an especially huge thank you to Louise for getting me through the darker times - you are my sunshine.

Summary

We investigate some problems commonly found when dealing with stereo images. Working within the context of visual effects for films, we explore software solutions to issues arising with stereo images captured on-set. These images originate from a wide variety of hardware which may or may not provide additional data support for post-production needs. Generic software solutions are thus greatly to be preferred.

This dissertation documents contributions in the following three areas. Each project was undertaken at Double Negative and investigated with the aim of improving the post-production pipeline for 3D films.

Colour matching is the process whereby the colours of one view from a stereo pair are matched with those of the other view. This process is necessary due to the fact that slight differences in hardware and viewing angle can result in some surprisingly large colour discrepancies. Chapter 3 presents a novel approach to colour matching between stereo pairs of images, with a new tool for visual effects artists given in section 6.2.

Vertical alignment of stereo images is key to providing a comfortable experience for the viewer, yet we are rarely presented with perfectly aligned footage from the outset. In chapter 4 we discuss the importance of correcting misalignments for both the final audience and the artists working on these images. We provide a tool for correcting misalignments in section 6.3.

Disparity maps are used in many areas of post-production, and so in chapter 5 we investigate ways in which disparity map generation can be improved for the benefit of many existing tools at Double Negative.

In addition, we povide an extensive exploration of the requirements of 3D films in order to make them presentable in the cinema. Through these projects, we have provided improvements to the stereo workflow and shown that academic research is a necessary component of developing tools for the visual effects pipeline. We have provided new algorithms to improve the 3D experience for moviegoers, as well as artists, and conclude by discussing the future work that will provide further gains in the field.

Chapter 1

Introduction

There are many papers on the topic of feature matching, optical flow, or tone matching images (see chapter 2), but the problem of colour mismatch between stereo images has received little attention. In some regards, the problems are simplified versions of ones that have been widely discussed. For example, the problem of finding feature matches between two images is easier to solve when it is known that the camera sensors are only separated by a matter of centimetres. This simplifies the problem from one where there may be large distances involved, such as in Elias [2007]. However, there are some problems that are made more complex by our constraints and require more thought in the solution. For example, ensuring the colours of every part of a stereo pair of images closely match their counterpart is a more complex problem than that of tone matching two different images. This is because when you view images separately, one after the other or even side by side, it is easier to "get away with" an imperfect match. When dealing with stereo, however, both views are observed at the same time and so local colour difference must be minimised in order for it to work – it is much harder for a viewer to fuse images if they are wildly mismatched. That said, there has been very little research into the threshold at which a viewer will experience discomfort given colour mismatched stereo images. For the purposes of this research, we work under the assumption that if we get to the point where we can lay a section of an image next to its stereo counterpart and perfectly align them, it should not be possible to observe any difference. (This kind of test is not applicable in situations where the differing angles of view of an object are so wide that it is impossible to align them in such a way that colour would be the only factor in noticing a difference, but it is a good way to test a lot of areas of a given pair of images where it is possible to do this.) We would encourage further research studies into the perceptual problems of stereo and the determination of some "threshold of comfort" when viewing imperfect stereo, as anecdotal evidence would

suggest that this can be a problem, but it is difficult to determine at what point the problem starts.

In addition to this we seek to improve Double Negative's vertical alignment toolset, by implementing various techniques that can be used in different situations when there is vertical misalignment between stereo images. It is our hope that in developing these tools we can improve the post-production pipeline and allow the creation of 3D movies with less effort from the artist being spent on matters that would be of no concern with a 2D movie.

We also investigate the relationship between stereo images at a lower level by looking into disparity map generation. The primary focus of this project is to explore ways to create a "cleaner" disparity map than those that are produced by most existing methods. We attempt to reduce the erroneous areas usually seen around edges in disparity maps to improve their overall quality and usefulness.

The tools that have been developed in the process of this research have produced several papers (section 1.1) and have been used on many film and TV projects, some notable examples of which can be found in section 1.2. In addition to these, two film credits have been awarded and can be seen in section 1.3.

1.1 Publications

Willey, S., Willis, P., Clifford, J., and Waine, T. (2014). Colour matching between stereo pairs of images. In *Computer Vision-ACCV 2014 Workshops*, pages 289–298. Springer.

Willey, S., Willis, P., Waine, T., and Hall, P. (2016). Localised colour matching between stereo pairs of images. In *Proceedings of the 13th European Conference on Visual Media Production (CVMP 2016)*, page 10. ACM.

Willey, S. A. and Keech, G. (2016). Warping with accumulated motion vectors. In *ACM SIGGRAPH 2016 Talks*, page 30. ACM.

1.2 Notable projects

The Pirates! In an Adventure with Scientists!. Dirs. Peter Lord, Jeff Newitt. Aardman Animations, Sony Pictures Animation, 2012. 3D Film

Exodus: Gods and Kings. Dir. Ridley Scott. Chernin Entertainment, 2014. 3D Film

Avengers: Age of Ultron. Dir. Joss Whedon. Marvel Studios, 2015. Film

Agent Carter. Prod. Sara E. White. ABC Studios, Marvel Television, 2015. Television

Fantastic Beasts and Where to Find Them. Dir. David Yates. Heyday Films, 2016. Film

BrainDead. Exec Prods. Robert King, Michelle King, Ridley Scott, David W. Zucker. Scott Free Productions, King Size Productions, CBS Television Studios. 2016. Television

1.3 Film credits

Ant-Man. Dir. Peyton Reed. Marvel Studios, 2015. Film

Miss Peregrine's Home for Peculiar Children. Dir. Tim Burton. Chernin Entertainment, 2016. Film

1.4 Contribution of this thesis

This thesis attempts to tackle a very real problem faced by visual effects companies today. The increased use of stereo rigs to capture 3D footage has led to a large increase in the post-production work needed to make a movie "theatre ready", since it means twice as many images must be manipulated for any given shot. One of the most time consuming tasks in the stereo pipeline is ensuring each view matches up so that the image can be fused comfortably by viewers. With a standard 2D movie, it is only necessary to make sure that visual effect elements blend well with the recorded footage and look good. If this is achieved then the viewer will be able to watch the

whole movie (whether they enjoy it or not is a completely different matter). With 3D films, however, it is necessary to take into account the process that the viewer must go through in order to even get as far as seeing the intended images.

This thesis provides contributions in three key areas. Firstly, we present an algorithm for localised colour matching between stereo images (chapter 3) which produces good quality results on high resolution images while preserving the original noise pattern of the corrected image. The industrial contribution in this area is presented as a Nuke plugin for full integration into Double Negative's existing pipeline, providing artists with a tool to produce results more quickly and easily than the manual method.

Another important consideration in making images comfortable to view is ensuring the images are correctly aligned vertically. We explore this problem in chapter 4 and contribute a time-saving tool in section 6.3 to provide artists with several possible methods of correction, including a method of grouping by x-disparity not provided by other existing tools.

We also investigate disparity map generation techniques and explore ways in which we might improve them (chapter 5). This is key to the stereo pipeline as disparity maps are used by many existing tools, and could be used by many more if the results they achieve were improved. We explore the current state of disparity maps both in academia and industry and look at the best ways to improve performance specifically within Double Negative.

Additionally, this dissertation contributes a detailed exploration of the requirements of 3D movies in a post-production context.

1.5 Motivation for work

Double Negative have a very strong Research and Development department, but with such a large user base (the company has grown to over 1000 artists in 4 countries) it can often be the case that developments are more reactive than proactive. With the growing popularity of 3D movies, it makes sense to expand existing toolsets to be able to cope with stereo image sequences. Double Negative are interested in using academic research as a means to creating software that will be beneficial in the creation of 3D movies, and would like to explore the impact of collaborating with academic partners. During the course of this research Double Negative have become a much more collaborative company, taking on further EngD students and taking

part in several cross-site projects such as the OAK[1] and ASAP[2] projects.

1.6 Stereo image capture

In this section we discuss the process of stereo image capture for the purposes of film production. While there are several different methods used in practice, a general overview of the problems that must be overcome by all of them is presented before going into the details of the various approaches. It is important to build up a solid picture of the techniques used to create the images that will be processed in order to understand the need for post-processing. Specific makes and models of cameras and rig equipment will not be discussed, rather the focus will be on the different categories of equipment that may be used.

1.6.1 Camera rig

There are two main types of camera rig, referred to here as a side-by-side camera rig and a beam-splitter camera rig. These rigs each provide a solution to the stereo recording problem but come with their own disadvantages and caveats.

The side-by-side camera rig is exactly as it sounds – two cameras are placed next to each other and synchronised so that they are recording at exactly the same time but from slightly different viewpoints. With this rig it is possible to alter the interaxial distance quite easily by simply moving the cameras along the bar connecting them. The interaxial distance is the distance between the sensor of each recording device. This is similar to the interocular distance which is the distance between the viewer's eyes.

The beam-splitter camera rig is a much larger rig and involves placing the cameras in such a way that one is facing forwards as usual and the other is positioned above, facing down; below, facing up; or to one side, facing perpendicular to the forward facing camera. In each case the second camera is positioned slightly ahead of the first and a beam splitter is placed at a 45 degree angle between the two. This beam splitter will then allow 50% of the light through to the first camera and reflect 50% of the light towards the second camera, in this way both of the cameras can capture the scene at the same time from slightly different viewpoints (figure 1-1). One big

[1] http://www.cs.bath.ac.uk/ pmh/OAK/OAK.html
[2] http://gtr.rcuk.ac.uk/project/54223401-C8D2-4154-8F63-9128F9CC6F1B

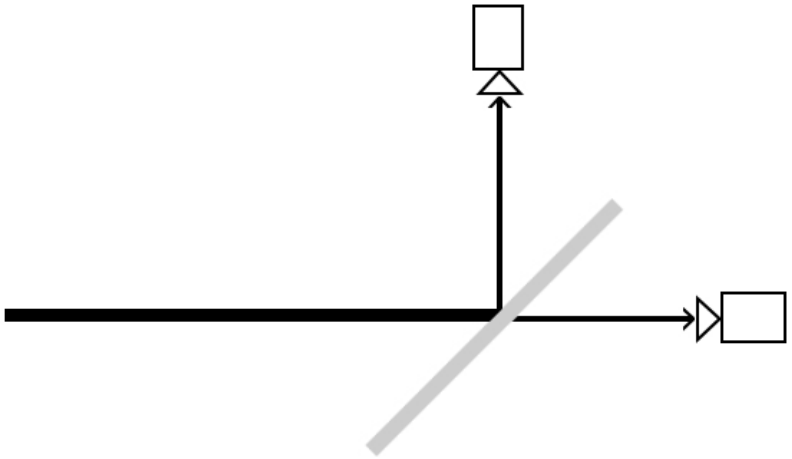

Figure 1-1: A simple beam-splitter rig.

advantage of this setup is the ability to reduce the interaxial distance to be much smaller than that of the side-by-side camera rig as there is no need to make room for the body of each camera. This is ideal for shooting close-up elements as too much of an interaxial distance would over exaggerate the 3D effect and create large occluded areas. As a general rule, the closer the object is to the camera the smaller the interaxial distance should be to create a comfortable 3D effect, for objects that are far away the interaxial distance must be increased in order to perceive any 3D effect at all. The disadvantage of this rig set up is that each camera will lose a stop of light due to the beam splitter only letting half of the light reach each one. One stop, or Exposure Value (EV), corresponds to a standard power-of-2 exposure step, used to indicate an interval on the photographic exposure scale. On this scale, an increase of one stop means half as much light being recorded, either through an increase in shutter speed or a decrease in relative aperture [Okun and Zwerman, 2010, p.357]. Another issue with this rig is that it is very large, either height-wise or width-wise depending on the method used. Due to this size it must be very rigid in order to minimise vibration and camera shake as each camera will be affected in a different axis so this will likely need to be stabilised during post production.

1.6.2 Camera orientation

Both of the above camera rigs have the ability to work with either a parallel or toe-in setup, these have some important differences which require careful consideration when deciding which to use. Parallel cameras are set up

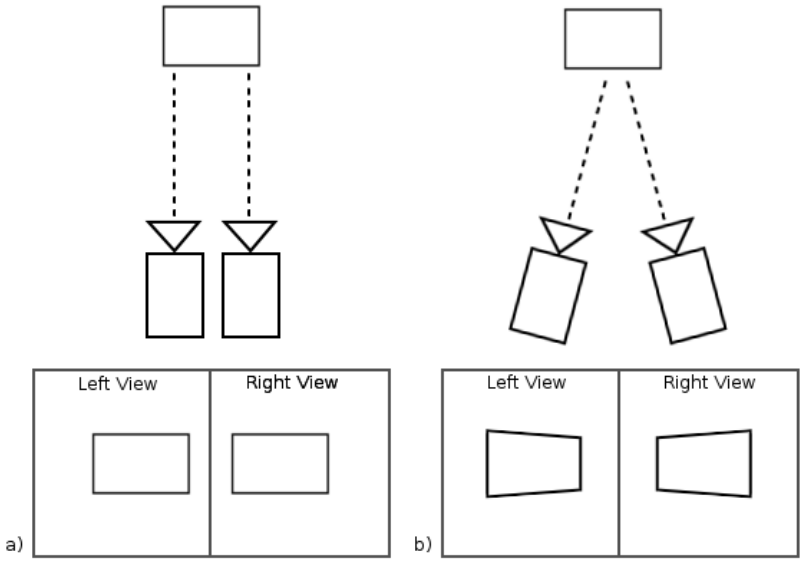

Figure 1-2: An example of a) parallel and b) toe-in setups with example image capture.

in such a way that the cameras are both pointing forward, parallel to each other. In this instance there is no point of convergence between the cameras and so everything in the scene will have negative parallax, meaning that every pixel will appear to be in front of the screen. In order to create a more comfortable viewing experience, this will need to be adjusted in post production so that the point of convergence is at such a level that the stereo image is easier to fuse.

Toe-in cameras, however, attempt to simulate the convergence of human eyes by being angled towards each other to adjust the convergence point of the cameras. This has the advantage of providing a pair of images that have an appropriate convergence point for the focus of the scene but these images can suffer from the issue of keystoning. Keystoning, or Keystone Distortion, is the name given to the effect achieved when photographing a plane in such a way that the camera is not angled perpendicular to it, thus making the nearer edge appear longer than the farther, this distortion would be equal and opposite in each image, making it difficult to fuse the images without additional processing (see figure 1-2).

In order to gain the advantages of the toe-in setup but avoid the keystoning effect, a process known as Horizontal Image Translation (HIT) is used. Using a regular parallel setup, the images can simply be moved in post-production in order to create a point of convergence wherever the director chooses, this results in some image loss at the edges where there is no corresponding area but this is usually not a big problem.

1.6.3 Viewing stereo footage

Once the stereo data has been captured, there are several options for the exhibition of the image or sequence. There are many subclasses of each category that won't be explained in detail as just an overview of techniques is necessary for some background information. The main viewing categories for 3D are anaglyph, active shutter and polarised. Anaglyph is the process by which one view is displayed in one colour, for example red, and the other view is displayed in another, for example cyan (examples of anaglyph images can be found in chapter 4). The idea is that one of the images will be filtered out by each lens resulting in just one image reaching each eye and thus creating a 3D image. This method has the disadvantage of not allowing colour information to be accurately passed to the viewer, although there are methods to get this information through that have been explored. One of these methods is to use narrow band interference filters whereby each lens allows red, green and blue through but at slightly different wavelengths, for example red will be allowed through at 629nm in one eye and 615nm in the other[(3)].

Polarised, or passive, systems simply have oppositely polarised lenses to separate the images into each eye. The projector used will have an electro-optical liquid crystal modulator in front of the lens to alternately polarise each frame. It will circularly polarise the frames clockwise for one eye and anticlockwise for the other. This technique requires a silver screen in order to maintain the light polarisation upon reflection as well as to minimise any light loss to counter the inherent loss of light by the polarising filters.

Active shutter glasses use LCD screens within the lenses to block out each eye in time with the image on the screen. When the screen is displaying the right view, the glasses will block out the left eye and vice-versa. These systems require batteries and are usually synchronised with the display using an infrared signal. They also require the display to work at a high frame rate of at least 120Hz – 60Hz for each eye.

1.6.4 Post-converting from 2D footage

Any fully computer generated effects which make use of a 3D model can be converted to stereo relatively easily. It is just a case of rendering the model from a slightly different angle. When live-action footage is involved, however, things become much more complicated. A widely used method for doing this is to take the single image that was captured and create

[(3)] http://www.dolby.com/uploadedFiles/Assets/US/Doc/Professional/Dolby_3D.pdf

3D geometry for each of the components in the scene, the characters for example. Then, deform this geometry in whatever way is necessary to get the correct look and re-project the original image onto it (see "Projection Deformer" in section 6.5). The results can then be captured from a slightly different angle by a second, virtual camera. The main alternative to this method is to rotoscope individual elements of a scene and group them into layers. These layers can then be horizontally offset by slightly different amounts to create a depth effect. The results of this method, however, will often have the negative effect of making each layer look flat which can be jarring for the audience (often referred to as a "2.5D" effect).

1.6.5 Limitations and scope

This dissertation comprises three related areas of research that each feed into and rely on each other. Colour matching, correcting vertical misalignment, and improving disparity maps between stereo pairs of images are all problems that require solving in order to create a good 3D movie experience. Although the cause for the problems described in this research are often hardware based, we will only be focusing on software methods to solve them. It is outside the scope of this research to develop a camera rig or system for recording stereo images. It is assumed that the hardware already in use is likely to be the standard for the foreseeable future and so any limitations presented are going to remain. Because of this, the focus will be on achieving high quality results with the images produced given all of these limitations.

It is also important to note that Double Negative have a very well established codebase and toolset, anything we create must be able to work alongside existing tools where appropriate.

1.6.6 Practical applications

Far from being a purely academic problem, the issues discussed in this dissertation are an important consideration in professional film production. At present, dealing with stereo footage is a time consuming process that causes the artists to spend more time just getting a movie up to a high enough standard for viewing comfortably and less time creating visually impressive graphics and effects. Minimising the need for manual work from the artist in this process will provide a significant improvement to the efficiency of post production facilities and artists wishing to create 3D sequences of images. If the problems of colour matching, vertical alignment, and disparity map

generation can be solved and integrated into the pipeline of a large post production facility then there are considerable cost savings to be made. 3D has become a major part of the cinema experience and although there has been some backlash to the format due to some examples of poor quality mono to stereo conversions being released, it shows no sign of disappearing at any time in the near future. Conversely, in fact, this has resulted in studios and filmmakers taking consideration of recording natively in 3D more seriously and has increased demand for such a system proposed in this report. Recent examples of projects that have used native 3D footage are "Avatar" and "Transformers 3: Dark of the Moon" (which included both natively captured and post-converted stereo footage). Examples which have made use of the tools we have developed are "Exodus: Gods and Kings", and "The Pirates! In an Adventure with Scientists!"

1.7 How we measure success

Dealing with stereo footage remains a problem in movie production and post-production, with many issues being dealt with using workarounds and adaptations of tools used for single view footage. Sometimes this is perfectly adequate, but at other times it results in the needless waste of artist time in order to achieve a high enough standard. We are focusing on the specific problems of colour matching, correcting vertical misalignment, and improving the quality of disparity maps between the two views of stereo footage, although solving these problems also provides tools for other areas. We have worked closely with artists, and received feedback at all stages of development to ensure we are providing a useful solution. In the absence of any sort of ground truth, we found this constant feedback during the development cycle to be invaluable.

The overall goal of this thesis is to make a significant contribution to the visual effects industry with cutting edge academic research. In light of this it makes sense to measure success based on the usefulness of the tools created. If a tool is used in the post-production pipeline in place of some existing method then we will count that as a success, since it would not be used at all if it were not an improvement on the previous methodology. If a tool is used and provides a significant time saving or quality increase then this would also mean success. It can be difficult to measure factors such as the quality of a colour match with some real world stereo footage since it is inherently subjective, and so we rely on feedback from the world class artists working at Double Negative to let us know when we're heading in the right direction.

Chapter 2

The state of the art

In this chapter we present a short survey of relevant techniques to give a background to the problems explored within this dissertation. There are three major projects that were undertaken, covering colour matching between images, correcting vertical misalignment between images, and investigating disparity map generation between stereo images. We therefore must examine the literature from many areas in order to develop a full understanding of the problems that we aim to solve. In the first instance, we must understand stereopsis and the human visual system. Through this understanding we can appreciate the need for all of the projects listed, since they are designed to aid in the binocular viewing of images.

2.1 Stereopsis

There is a large amount of research on the human visual system in several scientific areas. From biology, specifically vision science, to psychology and, of course, computer vision. We are unable to explore each area to its full depths, but we can provide an overview of the basic concepts. Wheatstone [1838] discussed the idea of the empirical horopter and the "fusable area" using an early stereoscope for testing. The horopter is the area within which points in 3D space will reach the same point on each retina, and therefore be fusable into a 3D image. This essentially means the field of view for comfortable 3D viewing. While this is referring to real-life scenes, it is still an important thing to understand for creating comfortably viewable 3D scenes on a computer (or cinema) screen. If, for example, we were to disregard the horizontal disparity threshold and create a stereo pair of images with a very large horizontal disparity, this will not have the desired effect of creating

more depth in the scene, but rather would render it completely unviewable [Ogle, 1952]. There is ongoing research [Istead et al., 2016] for dealing with situations where the captured disparity was too great, or otherwise unsuitable, for comfortable viewing.

It is also important to appreciate the difference between the ability for our eyes to compensate for some vertical disparity [Stevenson and Schor, 1997] when it is necessary in order to view a scene, and the viewer's capability to do so consistently throughout an entire feature length movie. It is clear that while some authors report an ability for subjects to fuse images with rather large mismatch problems [Ogle, 1955], these are all static images and not conducted over a long enough period of time to gather the necessary data on discomfort (in fact these did not even measure discomfort, only ability).

We have learned from studying the literature that the reason it is possible to fuse images with any level of vertical misalignment is due to a process known as cyclovergence [Hooge and Van den Berg, 2000]. This is the process by which each eye rotates in equal magnitude and in the opposite direction. This can help to correct for such alignment errors as would be caused by the keystone effect created from using a toe-in camera setup, for example. While this is a perfectly natural process that occurs frequently in everyday vision, it is still a major source of asthenopia (eye strain) in 3D movies where the degree of cyclovergence must be adjusted regularly or maintained for a long time.

There has been a lot of research into the discomfort that can be caused when viewing stereo movies [Hoffman et al., 2008; Yano et al., 2002, 2004], although these have largely been conducted on 3D TVs [Lambooij et al., 2011] with only a few assessing discomfort in movie theatres [Solimini et al., 2012]. Due to the fact that the viewing angle in a cinema is much larger, however, it is reasonable to assume that any issues found on the smaller display of a 3D TV would only be magnified on a larger screen. Also, the all encompassing nature of the view at a cinema means that there is no way for the viewer to gain some respite from the images, and so it is important to ensure that it is not too taxing for them in the first place. While watching a display at home, people will tend to alternate between viewing the display and viewing elements of the real world, but since this is not possible with an immersive experience such as a head mounted display it can cause issues for the viewer [Kooi and Toet, 2004]. The same is true in a cinema since the only viewable area in the room is the screen itself.

From exploring the literature, it is remarkable to learn all of the things our eyes do in order to compensate for errors in what we are viewing. It is particularly interesting, given the nature of this dissertation, to learn that all of the things we are trying to correct in 3D movies are already

corrected for by the eyes in some way, though doing so can quickly become overwhelming [Ukai and Howarth, 2008]. For example, chapter 3 focuses on colour matching between stereo pairs of images which is important for comfort during prolonged viewing. However, Ideses and Yaroslavsky [2005] assert that our eyes can actually accept a full resolution colour image in one eye, and a reduced resolution monochrome image in the other and still fuse it into a full colour, full resolution stereo image. Upon viewing the images in that paper it quickly becomes apparent, however, that the detrimental effect of viewing images such as these quickly makes it an uncomfortable experience. While this is an interesting revelation, it does not deter us from our goal of reducing eye strain through correction of colours between stereo images as we still believe this to be worthwhile and necessary.

In this section we have discussed the importance of keeping images within the fusable area for comfortable stereo reconstruction. Part of this involves not creating too much horizontal disparity between views, which largely comes down to on set decisions and is quite easily fixable through a translation applied to one of the views, but vertical disparity is a harder problem to prevent and solve. In chapter 4 we explore various methods of correcting vertical misalignment between stereo images in order to minimise the amount of work that needs to be done by a viewer's eyes in order to correctly view the images displayed.

2.2 Colour transfer

In this section we discuss the process of colour transfer between images, and some of the many techniques for achieving it. We begin by exploring some of the methods that have been developed for adjusting the whole colour palette of an image using another image, or some user interaction, as a reference. Following this, we look at more sophisticated approaches that allow for local changes where necessary and do not rely on a global solution to entirely fix the problem.

The problem of colour transfer between images could be solved in many different ways. The most basic solution would be to take the colour palette of one image and apply it to the other through, for example, histogram reshaping [Pouli and Reinhard, 2010]. Another option would be to first segment the image, then perform a more specialised colour match between each element as defined by the chosen segmentation algorithm. Alternatively, the desired result could be achieved on a per pixel basis through techniques such as optical flow [Lucas et al., 1981]. The main factors considered in choosing an appropriate solution are speed, accuracy and scalability. Speed is nec-

essary as this solution will be implemented into the pipeline of real world projects and must therefore not have any negative effects which outweigh the positives of such an implementation. Accuracy is essential as we can't leave the artist with more work to do than was needed before performing the colour match. And, finally, scalability is desired as VFX are having to be applied to footage with ever increasing resolutions. These range from the standard 2K (2048x1080), to full IMAX resolutions of 5.6K or even 8K [Okun and Zwerman, 2010, p. 467, 508].

HaCohen et al. [2013] compute dense correspondences in order to calculate the necessary colour change to bring about some regularity within a collection of images. However, the actual changes made are applied globally to each image in order to avoid many of the complications brought about by making local alterations. Due to the nature of the problem that this paper is attempting to solve, the solution found is very effective. It can be quite jarring to have a set of photos from the same time and place which have varied palettes due to exposure or white balance settings, for example. The authors are able to solve this problem with global changes to each image, since it is only the "look" of the images that need to be the same. In addition, the images typically used for their experimentation are merely taken in the same area, they are not necessarily taken of the exact same thing from a slightly different angle, and are certainly never intended to be viewed at the same time as a stereo pair.

With the specific problem of colour matching between stereo pairs of images for the purposes of movie production, there have been very few studies. One relevant study by Neundorf et al. [2012] attempts to find a suitable method for evaluating colour differences in stereo pairs of images. From the conclusions of this paper, we confirm our suspicions that the results of any algorithm we develop will have to be subjectively evaluated by the artist using the tool and it is unlikely that we could create a reliable "self-evaluating" colour correction algorithm. This paper also highlights the fact that no studies have been published which thoroughly analyse the extent to which colours can differ between stereo image pairs before causing discomfort to the viewer.

When dealing with colour adjustments over video sequences, rather than still images, it is important to apply some temporal smoothing, as highlighted by Bonneel et al. [2013]. In this paper we see that sudden changes of scene content, such as a new character walking into frame, can have a big effect on the colour adjustment algorithm and need to be accounted for.

Reinhard et al. [2001] discuss the idea of converting between colour spaces in order to aid the transfer process. Due to observing that all three channels often require modification in RGB space, it is suggested that using one

with less of a correlation between channels would be advantageous. This is something that we may have to consider for our colour transfer algorithm, although the industrial constraints may limit our options in this area.

To overcome the issues we face with the stereo problem (see section 3.2), it will be necessary to make many local changes and a global solution will not be sufficient. This is mainly due to the fact that the images that are colour matched will be viewed simultaneously, and there is no way that a global solution can deal with all of the elements that could appear in a typical scene. For example, reflections, highlights, and iridescent materials will all create localised areas of colour difference that simply cannot be corrected by a global solution without seriously impairing the correction in other areas of the image.

There was some consideration given to the idea of performing a global correction and then just using local changes for areas identified as requiring them. However, a typical shot from a movie is likely to have several areas that will require local changes, and so the benefits of performing a global correction first are likely to be minimal. In addition to this, the added complexity of identifying any areas for which a global correction would be unsuitable, in order to exclude them from the global correction calculation, would mean an added investment in time both for development and running of the algorithm. In light of this, it was decided that the best course of action would be to focus on making localised changes throughout the entire image and therefore deal with particularly troublesome areas simply as a matter of course, and without the need for identification beforehand or special consideration during the correction. Chapter 3 details further exploration of the colour matching problem as well as a solution which provides a high enough quality to be of use in the post-production pipeline.

2.3 Calculating disparity

In this section we discuss some of the various methods for finding the corresponding areas in stereo pairs of images. We will explore different techniques available for doing so with both pre-segmented and non-segmented images. It is likely that pre-segmented images will offer a greater degree of accuracy when finding correspondences but the additional processing time may make the segmentation stage unfeasible.

2.3.1 Optical flow

Introduced by Horn and Schunck [1981], and utilised many times since, optical flow is a popular technique for tracking the movement of pixels across frames. There are several methods of implementing optical flow and in-depth surveys of the various techniques can be found in Beauchemin and Barron [1995] and Fleet and Weiss [2006]. The main problem with this technique, for our purposes, is that it can be very computationally expensive, especially with large resolution images. There is also a concern regarding the temporal consistency over a sequence of images if calculating for every frame. This is discussed by Lang et al. [2012] though the solution found greatly simplifies the problem and suffers a loss of accuracy in favour of efficiency.

Although optical flow is usually used to find the movement of pixels from one frame to another across time, we can treat one view in the stereo pair as frame zero and the other view as frame one and imagine a static scene with a moving camera. Thus optical flow can be used to find the correspondence of individual pixels between the two views of a stereo pair.

The Middlebury Dataset [Scharstein and Szeliski, 2002] offers a good way to benchmark disparity map generation techniques and there are many optical flow algorithms on the leaderboard. Many of the images in this dataset are too low resolution for our purposes, but there have recently been some higher resolution images added [Scharstein et al., 2014]. It is worth noting that although these images are higher resolution than previous benchmarking datasets, at 6k they are lower resolution than what we might reasonably expect to have to deal with for a blockbuster movie. Also, this dataset includes only stationary images and so good performance with these may not necessarily mean an algorithm performs well with stereo sequences. Sun et al. [2014b] implements several techniques and compares them not only against the Middlebury Dataset, but also the MPI Sintel [Butler et al., 2012] and the KITTI [Geiger et al., 2012] datasets.

At Double Negative, an in-house tool has already been developed and used for many years based off the Bouguet [2001] implementation of the Lucas et al. [1981] approach. This tool produces good results, but there is definitely room for improvement. In fact, this tool has largely been replaced by the VectorGenerator[(1)] node in Nuke which produces high quality results. Unfortunately we are unable to find out exactly what algorithm this tool uses due to its proprietary and commersialised nature. While this tool is good for many uses in visual effects, we feel that there are improvements to be made and so have continued to explore potential ways we might do so.

[(1)] http://help.thefoundry.co.uk/nuke/8.0/content/user_guide/kronos_motionblur/vectorgenerator.html

One concern when creating disparity maps using optical flow is the difference between the *motion field* and the *apparent motion* [Baker et al., 2011]. It is more or less accepted that this will be an issue regardless of our solution and it is deemed acceptable that artists may have to perform extra processing on areas which, for example, contain reflections that appear to move in the opposite direction to the reflective object itself.

Optical flow estimation remains an active area of research with many papers published in recent years. Many continue to build directly off of the classic Horn-Schunk method, such as Chen and Koltun [2016] who use full regular grids to initialise continuous interpolation as a method to improve its performance. Sun et al. [2010] show, with the introduction of *Classic+NL*, that classical methods still perform well when implemented using modern practices. Building further on the *Classic+NL* method, using a channel representation and a descriptor constancy assumption rather than the usual brightness constancy assumption offers an alternative to classic gaussian pyramid techniques [Sevilla-Lara et al., 2014].

Our main concern with regards to improving the current state of disparity map generation techniques is the edges within an image. All optical flow techniques perform somewhat erratically at the borders between items in a scene, and we hope to find a way to improve on this. Sparse-to-dense interpolation shows promise for preserving edges, improving on current coarse-to-fine methods [Revaud et al., 2015].

Another area of exploration is in the use of multiple frames to calculate optical flow across a sequence. Since simply calculating the flow between 2 consecutive frames can often be insufficient [Sun et al., 2012], Ochs et al. [2014] calculate point trajectories for long frame ranges which are less susceptible to short-term variations. Sun et al. [2013] formulate a fully-connected layered model for the calculation of scene segmentation and motion estimation together across long frame ranges.

Alternative methods of motion estimation include using object class labels to determine the appropriate motion model to apply in each region [Sevilla-Lara et al., 2016], or using convolution neural networks as a supervised learning task [Dosovitskiy et al., 2015]. There are certain cases where a specialist motion estimation technique may be required, such as with the flow of fluids [Doshi and Bors, 2010] or non-rigid deformations [Garg et al., 2013], but for our purposes these will be unnecessary in most cases.

In addition to this, we are on the hunt for a technique that will be able to cope with the extremely high resolution images that we regularly need to deal with, since many optical flow techniques are too slow, or even entirely incapable of dealing with such images.

2.3.2 Image registration

Image registration is the process of matching two images that are taken, for example, from different viewpoints, sensors, or times [Brown, 1992]. In the case that we are interested in, the images are taken at exactly the same time (camera synchronisation is a key assumption), from slightly different viewpoints (usually only a few centimetres) with physically different, but in every way identical, sensors. There are various published techniques that deal with image registration and both Brown [1992], and Zitova and Flusser [2003] provide comprehensive surveys for image registration techniques up to 2001.

With pre-segmented images, registration techniques could be used to make very accurate colour changes as each area can be matched to its corresponding segment in the other image. With a non-segmented pair of images, however, there would be an issue with scene elements at different depths and so these techniques would not be usable on their own. Using some algorithm such as SIFT [Lowe, 1999] to find key points in each image could, however, be used for the vertical alignment process when combined with image registration techniques.

2.3.3 Structured lighting

A popular method of calculating disparity is to project structured lighting onto a scene in order to measure the distortions and convert them into disparity information [Batlle et al., 1998; Chen et al., 1997; Horn and Kiryati, 1999]. This technique can produce some very high quality results [Scharstein and Szeliski, 2003] and is, in fact, the method used for ground truth in the latest Middlebury Dataset[(2)]. This is still an active area of research [Salvi et al., 2010], with accuracy constantly improving. However, it is not a technique that we are able to use for two reasons. Firstly, the nature of this technique only really lends itself to static scenes, since to gather the RGB data you must have the light projection turned off, and to gather the depth data you must have it turned on. Secondly, even if the projected light were invisible, such as the infrared light used by the Xbox Kinect [Freedman et al., 2012], movie makers are generally nervous about anything being projected onto a live set in case of any interference. After some brief investigation of the Xbox Kinect it was found that even without the limitations already mentioned, it would be unsuitable for the task at hand due to the fact that it is designed to be used in a fairly controlled environment and accuracy quickly drops over large distances [Khoshelham, 2011].

[(2)] http://vision.middlebury.edu/stereo/data/2014/

2.4 Other notable research

Through the course of this research the idea of segmenting the images in some way was explored on several occasions. This section covers the areas that were looked at during this exploration, though it soon became apparent that these methods would not be suitable for what we were trying to achieve and so this path was abandoned in favour of other options.

We will briefly explore some of the different methods for image segmentation, including the use of classification models [Ren and Malik, 2003], non-rigid image registration [Crum et al., 2004] and graph cuts [Kwatra et al., 2003], and assess their viability. For a more detailed survey of image segmentation methods, see [Zhang et al., 2008].

2.4.1 Thresholding

Using a threshold value is the simplest method of segmentation as it is only concerned with intensity values of individual pixels and does not take any spatial information into consideration. A cut-off value is specified and any pixel with an intensity on one side of this value becomes one segment while anything on the other side goes to the other segment. This can be a very useful technique for problems where the background and foreground of an image are well defined, for example when dealing with black text on a white background. A detailed survey of various thresholding techniques can be found in the paper by Sezgin et al. [2004].

There are many reasons why a basic thresholding technique would be unsuitable for the problems presented. At the very least, extra parameters would have to be imposed so as to take spatial information into consideration, but even then the diversity of elements found in any given shot of a Hollywood movie would prove too much for this solution to deal with. This technique is better suited to occasions when there are very few colours in a scene and areas of the same segment are not necessarily spatially co-located, such as in the example mentioned above where black text must be separated from a white background.

2.4.2 Clustering

Cluster based methods of image segmentation attempt to group pixels together that have some common properties, for example, colour similarity or

proximity. Ren and Malik [2003] offer an example of the use of Gestalt theory in creating a cluster-based segmentation. A full explanation of Gestalt theory can be found in Wertheimer [1938] but the basic principles for grouping pixels together are as follows: proximity, similarity, common fate, continuity, closure, and foreground articulation. Many of these principles are common sense and this method largely describes the subconscious process we go through when seeing things in the real world.

The k-means method for segmentation divides an image into clusters based on initial points such that the sum of squares within each cluster is minimised [Hartigan and Wong, 1979]. This means that the distance, by some measurement (colour similarity, physical location etc.), of each point in a cluster to the cluster's centre is minimised through moving the centres and moving points between clusters until an optimal value is found for all clusters and points. This involves creating cluster points, finding which pixels belong to which cluster through some method and then merging clusters as necessary. This often requires user input at the start of the segmentation either to select the number of clusters that will be created or even to manually specify the cluster points. There have been some attempts at automating these processes [Ray and Turi, 1999] but the k-means method has other problems associated with it. If the cluster points are not placed correctly or the wrong number of desired clusters is entered then the algorithm may converge on a local minimum and yield incorrect results. This problem would be made worse with the use of stereo images as random placement of cluster points would be likely to create differing segmentations; manual placement of cluster points would require too much input from the artist; and trying to place cluster points in the second image based on those in the first with some approximation of shift would require an extra pre-processing step that will simply add to the time penalty of using this method.

2.4.3 Classification Model

Ren and Malik [2003] introduce a classification model whereby segments are classified as "good" or "bad" based on human-segmented images which are then held in a database [Martin et al., 2001]. This method firstly requires a lot of input to create the human-segmented reference images and secondly will require a lot of space for the resulting database containing all of the images with reference segmentations. In Ren and Malik [2003] the authors created a simple algorithm for segmenting 240x160 px greyscale images and just searching through the database took up to 30 minutes. As has previously been mentioned, the images we need to be able to handle are full colour and can exceed 10,000x8,000 pixels. As such, it is unfeasible to use such a solution as the time constraints and level of interaction needed are

far too great. In the aforementioned paper, however, they do make use of the normalised cuts algorithm which may be useful to solving our problem.

2.4.4 Graph Cuts

The use of graph cuts in image segmentation is very popular and there are many examples of good results being achieved [Kolmogorov and Zabih, 2006; Shi and Malik, 2000]. The basic idea with these techniques is to view the image as a graph of pixel values and make "cuts" where the energy required to do so is minimised. One example is to impose a penalty for assigning a cut position which essentially ensures that cuts will only be made when absolutely necessary and so avoid excess processing. Several Graph Cut solutions make use of the Potts model [Boykov et al., 2001] which does exactly this.

One major drawback of the Graph Cut technique is that scalability quickly becomes an issue. Many of the publications mentioned use sample images of a much lower resolution than those that are used in VFX production. It is also best suited to binary images, or images where few labels are necessary, and becomes less viable with greater complexity. The Graph Cut technique can produce some very good results when facing simple pixel labelling problems with not too much complexity. However, a typical scene in a movie production will have several elements that need to be distinguished from each other in order to correctly add the necessary visual effects. In this instance, a Graph Cut technique would be unsuitable.

2.5 Conclusions

While there are very few items of literature that are directly related to the problems we face specifically for creating 3D movies, there are certainly many that deal with problems we must solve on the way. Colour matching, vertical alignment, and disparity map generation are the areas that we have chosen to focus on, and the rest of this dissertation shows our efforts in improving Double Negative's tools as well as the overall state of the art. 3D movies are playing a bigger role than ever in box office revenue and it is important to get the techniques right in order to provide the best possible experience to the movie going public. From investigating the literature we can see that many of the areas we are looking into are very active areas of research, and so there is plenty of prior work to build on in order to reach the solutions we need to our particular set of problems.

We conclude that while much of the research being conducted in these areas are not concerned with 3D movie production, they still offer a solid foundation from which to work from. We aim to utilise this prior work in a way that will create the most useful possible tools for a working post-production pipeline, with all of the constraints and limitations that this entails. Many of the papers explored here do not have such limitations as, for example, not having any control over the hardware used in image capture, or having to create tools that work alongside many others that have been created over a period of nearly 20 years in the Research and Development department of Double Negative.

Chapter 3

Localised colour transfer between stereo images

In this chapter we present a technique for colour matching between stereo pairs of images. We use a pre-computed disparity map to find corresponding areas and achieve high quality results for localised colour grading. We avoid direct pixel-to-pixel matches and instead use a gridding system in order to constrain the area of focus and allow for a more subtle manipulation of colour. Our method maintains existing noise without introducing more, allowing it to be used in movie post-production without affecting other artistic decisions. We bring together elements from image correspondence algorithms and colour transfer techniques in order to manipulate localised areas to have a very close match.

3.1 Introduction

We discuss the problem of matching the colours of one image to those of another, specifically two images which together form a stereo pair. This problem differs from many other colour transfer problems in that it is very important that specific areas are as closely matched as possible. Other problems, such as those presented by Senanayake and Alexander [2007], or by Grundland and Dodgson [2005] and Pitié et al. [2007] are trying to match the *tone* of another image, which can be achieved through a global solution with some histogram reshaping [Pouli and Reinhard, 2010], for example. Others use local colour changes to try and match the colour palette across an entire photo collection [HaCohen et al., 2013], but these images will still be viewed one at a time and so slight incongruity can go unnoticed

by a casual viewer. Our problem differs from these in the respect that two images will be viewed together at exactly the same time, fused into one stereo image, and so any local differences will cause problems for the viewer. When the colours do not quite match between stereo images, the viewer may not be able to pinpoint the exact issue with the image, and should in fact still be able to fuse the images fairly well, but areas of the image will "pop" and give the viewer the sense that something is not quite right. Over the course of a feature length movie's running time this can result in headaches and fatigue, ruining the viewing experience.

3.2 Problem description

Colour disparities can occur for many reasons when recording stereo images. When using the beam-splitter rig, as shown in figure 1-1, there can be a problem of light polarisation. The beam is split by allowing certain polarisations through while reflecting others. This will result in problems where there are specular reflections, for example. In the most extreme of cases unpolarised light can hit a surface at Brewster's angle, meaning that the reflected light will be polarised parallel to that surface. This could result in the highlight or reflection being more pronounced in one of the images in the stereo pair (figure 3-1).

Another issue that can affect the colours of each image independently is the hardware itself. If the cameras are not perfectly calibrated before recording then there can be slight discrepancies in the colour palettes of the images captured. It could be assumed that in this instance there would simply be a constant and consistent colour change throughout each image in a sequence. This is often not the case, however, and some localised changes are still needed to match the images. Even if two cameras are perfectly calibrated, there is always an error tolerance in manufacturing which could result in subtle variation between units.

One side of each image in the stereo pair offers a specific case of the colour matching problem. For each image there will be one side with some information that is not seen in the other view. This section of the image cannot be properly colour matched, but can be disregarded as it will have to be removed entirely in order to maintain the 3D effect. This is because 3D only works when a pair of images can be *fused* by the viewer, and an area of one image without a counterpart in the other cannot be fused. If these areas were present in the final output, they would result in the 3D image having fuzzy left and right borders. By removing these sides, a clean-bordered window is maintained through which the audience can view the images, and

Figure 3-1: An example of the same shot taken with differing orientations of a polarising filter.

the overall experience is improved.

This project is restricted to post-production solutions to these problems, due to the inaccessible nature of a film set and the fact that these images originate from a wide variety of hardware. Further restrictions resulting from the fact that our solution is to be used for major film productions are as follows:

- Noise must be preserved. Noise plays an important part in the final look of a film and so any smoothing would be a creative decision that would be dealt with in a separate part of the pipeline.

- We must be able to alter the colours of *either* the left or right view. When filming a 3D movie, there will always be one view that is deemed the "hero" that the other view should be matched to. (Note that the "hero" view can switch between different shots on the same movie project and so we must allow the artist to be able to select this at the time of processing.)

- Our algorithm must be fast enough and accurate enough to offer an overall improvement to the post-production pipeline.

3.3 Background and related work

The problem of colour transfer between images could be solved in many different ways. The most basic solution would be to take the colour palette of one image and apply it to the other through, for example, histogram

reshaping [Pouli and Reinhard, 2010]. Another option would be to first segment the image, then perform a more specialised colour match between each element as defined by the chosen segmentation algorithm. Alternatively, the desired result could be achieved on a per pixel basis through techniques such as optical flow [Lucas et al., 1981]. A thorough survey of the various techniques of colour correction is provided in [Faridul et al., 2014], including the different applications of the various algorithms and the limitations associated with each.

The authors of [HaCohen et al., 2013] compute dense correspondences in order to calculate the necessary colour change to bring about some regularity within a collection of images. However, the actual changes made are applied globally to each image in order to avoid many of the complications brought about by making local alterations. Due to the nature of the problem that this paper is attempting to solve, the solution found is very effective. It can be quite jarring to have a set of photos from the same time and place which have varied palettes due to exposure or white balance settings, for example. The authors are able to solve this problem with global changes to each image, since it is only the "look" of the images that need to be the same. In addition, the images typically used for their experimentation are merely taken in the same area, they are not necessarily taken of the exact same thing from a slightly different angle, and are certainly never intended to be viewed at the same time as a stereo pair.

To overcome the issues we face with the stereo problem, it will be necessary to make many local changes and a global solution will not be sufficient. This is mainly due to the fact that the images that are colour matched will be viewed simultaneously, and there is no way that a global solution can deal with all of the elements that could appear in a typical scene. For example, reflections, highlights, and iridescent materials will all create localised areas of colour difference that simply cannot be corrected by a global solution without seriously impairing the correction in other areas of the image.

In [Hasan et al., 2011], the authors focus on the problem of reducing false positives and false negatives in the geometric feature correspondence detection process in order to improve their colour correction method. Interestingly, the aim of this paper is to perform the colour correction phase as a first step in order to improve disparity map generation. Our technique differs from this in two important ways. Firstly, we find that our disparity map generation is not significantly affected by the slight colour differences we are aiming to correct and so we create the disparity map first and use its results for our colour correction algorithm. Secondly, we favour the removal of bad pixels over the retention of good pixels and so remove more data as a precaution.

In [Neundorf et al., 2012] the authors test and compare several colour matching algorithms, as well as attempt to find a good metric by which to measure the success of these algorithms. Of the four methods developed, however, only one provided localised colour changes. This method was found to be insufficient both in terms of overall result and time taken to achieve it. In this paper, the authors also investigate several possible methods of determining the level of success of a given colour matching algorithm, but did not find a consistently accurate way to measure colour correction success when compared to subjective evaluation.

3.4 Contribution

This chapter's outlined contribution is to a technique which brings together elements from image correspondence algorithms and colour matching techniques to adjust the colours of one image to match those of another. Where this differs from other work outlined above, is that it is dealing with two simultaneous images of the same scene and requires localised areas to have a very close match. Other techniques alter the entire image with a global solution which results in a mismatch between some areas upon closer inspection. We present an algorithm which can accurately adjust the colours of one image to closely match those of another, with localised adjustments creating very close colour matches for corresponding areas. We preserve the noise of the adjusted image, while disregarding this noise from the colour correction calculation. In addition, the industrial contribution made by this work is discussed in section 6.2, where we present a tool (EyeMatch) which is easy to use and able to produce results much faster than the manual approach.

3.5 The algorithm

We introduce a method to alter the colours of a selected image in such a way that noise and edge information are fully preserved. We use disparity information to find corresponding sections of images which are then ordered within each colour channel such that proceeding calculations are performed on an ordered list of pixels within a section, rather than on directly corresponding pixels. In doing this we mitigate the effect of noise in the original images and the disparity map. Figure 3-2 highlights the main steps of our algorithm.

Our algorithm takes as input the original left and right views, as well as a disparity map which we use to find corresponding pixels. It is possible to use these inputs to correct either the left or the right view, depending on the "hero" view of the sequence, but for simplicity we will simply describe the method of correcting the colours of the right view. Similarly, the disparity map can be used to find corresponding pixels in either direction but for simplicity we will start with the left view and use the map to find corresponding pixels in the right view.

Our disparity map (D) stores the values (u, v) to be added to the coordinates (x, y) of a pixel in the left view (L) in order to find the coordinates(x', y') of the corresponding pixel in the right view (R).

$$(u, v) = D(x, y) \tag{3.1}$$
$$(x', y') = (x + u, y + v) \tag{3.2}$$
$$L(x, y) \widehat{=} R(x', y') \tag{3.3}$$

We deal with each colour channel separately so each pixel in L and R must be split into 3. In this example the three colour channels are red (r), green (g), and blue (b). All of the following operations must be performed on all colour channels, though for brevity we will only show the red-channel calculations.

3.5.1 Divide each image into grids

In order to localise colour changes, we split each image into rectangles with dimensions provided by the user. We deal with each grid section separately so we have $0 \leq i < W$ and $0 \leq j < H$, where W and H are the width and height, respectively, of each grid section as given by the user. There is a special case at the right-most and bottom-most edges of the images where the height and/or width are often limited by the number of pixels remaining rather than the user specified values since it is rare for the images to be of dimensions perfectly divisible by the specified grid size (and it is not feasible for us to impose this restriction). The grid size can be specified by the user, and does not need to be defined as square since height and width can be adjusted independently. This is important since there is always a trade-off between desired smoothness and required precision. A smaller grid size will result in a greater precision (as long as a good quality disparity map is provided as input) but may produce a less robust result as a smaller area will mean fewer pixels being looked at when deciding what constitutes an

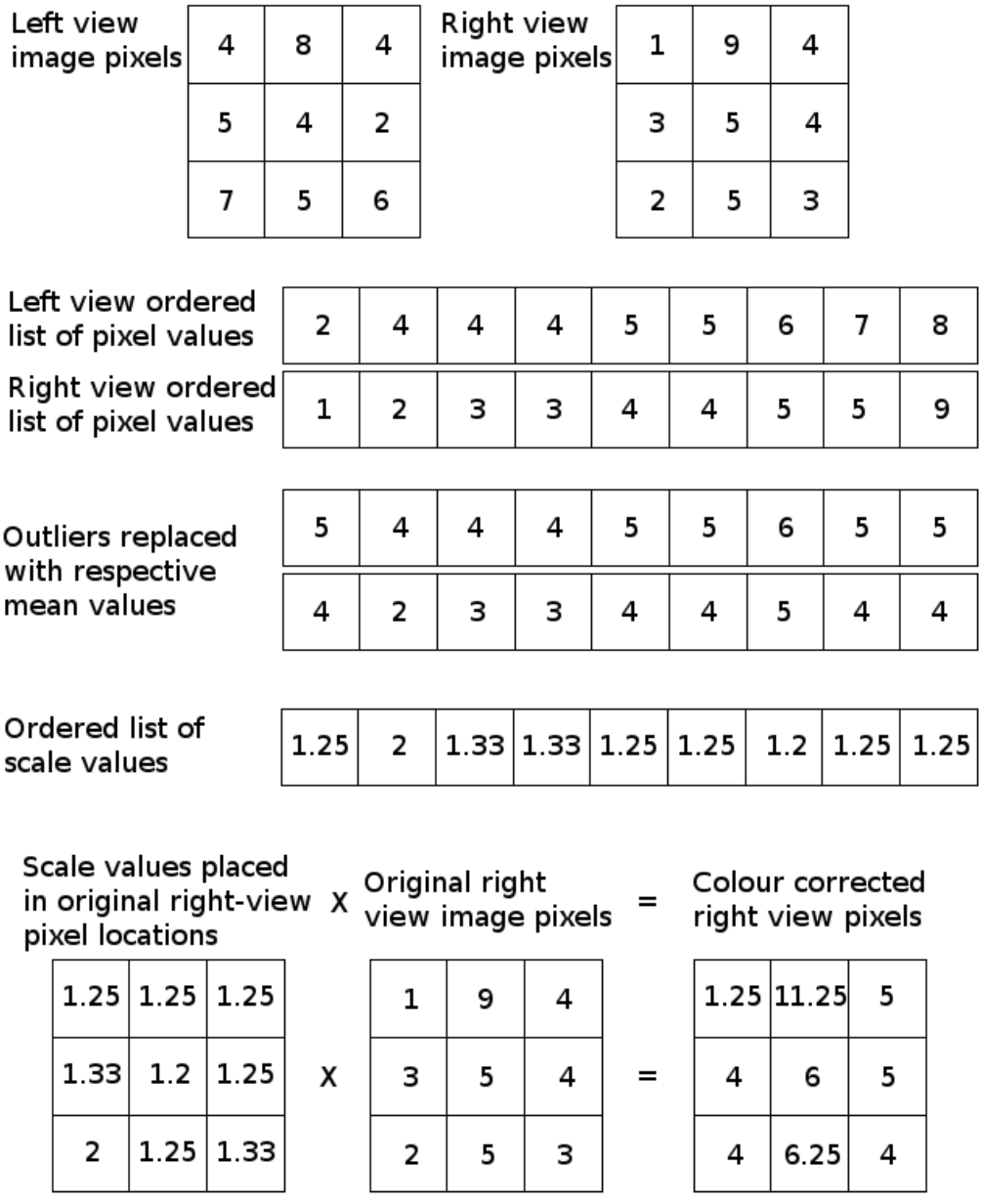

Figure 3-2: The basic steps of our colour matching algorithm.

outlier. This means there is more chance of some noise sneaking through. Conversely, a larger grid size will result in a smoother result but could result in some errors around edges, especially those with large occlusions.

3.5.2 Order values within each grid section

Our next step is to convert our single colour channel 2D arrays of pixels into ordered vectors. We store the original position of each value so it can be replaced at the end of the process. We now have an ordered vector (V) of length ($H \times W$) for each grid section, colour channel, and view. This ordering is important for step 3.5.4 since we do not want to compare pixels of exact corresponding co-ordinates in each image, but rather the pixels of the same rank within corresponding sections of the images (e.g. the lowest red channel value in a section of the left image is compared to the lowest red channel value in the corresponding section of the right image). This ensures the smallest possible change in colours which can still yield good results.

3.5.3 Calculate mean and standard deviation of pixel colour values and remove outliers

We next calculate the mean (μ) and standard deviation (σ) of each of the ordered vectors so that we may remove outliers which are likely to be noise. We ideally want as few noisy pixels as possible affecting the colour change calculations since the noise recorded by different hardware can vary greatly and will not be representative of the actual colour we wish to match but will seriously affect the outcome if included. For each colour channel (r, g, b), both the left and right views are checked for outliers and if any are found then they are changed to the mean value for each view. In our experiments we found a good threshold to be $\mu \pm \sigma$ as this cuts out a lot of the noise but preserves enough "good data" for the rest of the calculations required.

$$\begin{aligned} \text{If } ||V[k]_r - \mu_r|| &\geq \sigma_r \\ \text{then } V[k]_r &= \mu_r \\ \text{where } 0 \leq k &< H \times W \end{aligned} \tag{3.4}$$

The same is done for the green (g) and blue (b) channels.

3.5.4 Divide ordered pixel values

Once the outliers have been removed and replaced with mean values for all colour channels, each value in the left view is divided by its corresponding value in the right view to calculate the scale value (S) that will be used to alter the right-view colours.

$$S[k]_r = \frac{V^L[k]_r}{V^R[k]_r} \tag{3.5}$$

In the event that the values in the right-view vector are zero, then this calculation is not performed and the scale value for that pixel is set to 1 (no change). It is hard to imagine an eventuality where this would occur, however, since even filming total blackness will result in values greater than zero.

3.5.5 Return scale values to original image pixel order

Once calculated, each scale value is then placed in the original position of the pixel it refers to (in the example used throughout this section, the pixels in the colour change map will be arranged according to the original positions of the right view pixels), using the location recorded during the sorting process. We now have a 2D array with 3 colour channels that can be multiplied by the original right-view image to more closely match the colours of the left view. This colour change map is blurred slightly to reduce any existing noise or tiling artifacts before being applied to the right view. It is important to note that blurring the colour change map will have absolutely no smoothing effect on the actual image that it is later applied to – our algorithm maintains the fidelity of the images it alters. Now, where the viewer might have been uncomfortable viewing images L and R together, there will be no problems (at least where colour is concerned) in viewing L and SR together.

3.5.6 Reduce flickering in video sequences

When using this algorithm for video, there is one additional step that is required in order to create an acceptable output. We run each frame through the algorithm individually before performing a temporal blur between the colour difference maps for each frame. This works in the same way as the

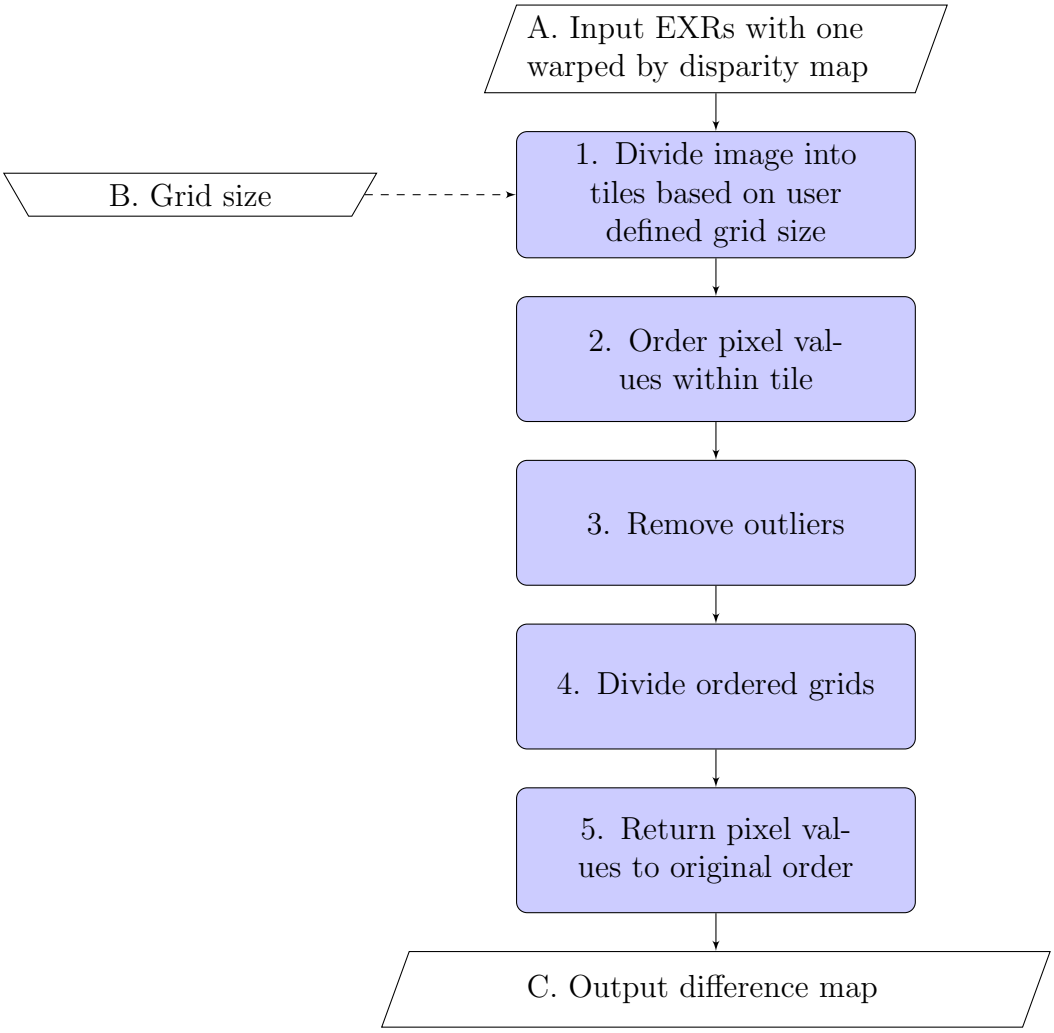

Figure 3-3: Flowchart for our colour matching algorithm

standard blur used to smooth the colour change map for a single frame, except that rather than using information from spatially grouped pixels it uses the same pixel from each adjacent frame. This process eliminates the "flickering" which can otherwise occur between frames when the necessary colour change calculated for a frame is different to the next.

The flowchart in figure 3-3 offers a simplified outline of the entire process.

A The input to our algorithm is two RGB images. One will be the original image of one view and the other will be the warped image of the other view.

B The grid size is specified by the user.

C The output is a difference map which shows by how much a pixel's colour values need to be multiplied to achieve an appropriate colour match.

1 Gridding the images into appropriately sized tiles allows the problem to be broken down into manageable chunks and allows different areas to be processed in parallel. The grid size is important, too large and the resulting output will be too blocky, too small and it will be too noisy.

2 The pixel values in each tile are first put in order, with a record kept of their original positions. This must be done for all colour channels.

3 Once the pixel values are ordered it is easy to calculate the standard deviation and remove any outliers. Any pixels found to be an outlier at this stage have their value changed to the mean of all pixels in that tile.

4 Once the tiles have been processed in the way outlined above, each pixel from the tile from the view which requires adjustment is then divided by the corresponding pixels from the other view. This provides a value by which to multiply the original image to achieve a colour match.

5 The pixel values must be returned to their original position within the tile to complete the process and produce the required difference map.

3.6 Results

Figures 3-4 and 3-5 show the original images as captured on set with a beamsplitter stereo rig. Close examination of these images shows discrepancy between colours which we aim to remove.

Figure 3-6 shows a checkerboard composite, where alternating squares show information from the left and right views, of the stereo pair of images shown above. Cropped versions of these images highlight the differences between colour more clearly and figures 3-7 and 3-8 demonstrate the issues faced.

It is not necessary to have performed alignment before colour matching,

Figure 3-4: Left view input image from a beamsplitter rig.

Figure 3-5: Right view input image from a beamsplitter rig.

Figure 3-6: By viewing the left and right views in alternating grid squares of the same image, it is easy to see the colour differences present throughout.

Figure 3-7: A closer look at a portion of the scene with a fairly uniform colour. Alternating squares taken from the left and right view highlight the difference between views.

Figure 3-8: Another close-up view, with alternating squares taken from the left and right views, showing the colour differences that must be corrected in another portion of the scene.

Figure 3-9: Disparity map showing x-disparity values for left view. These values are used to locate corresponding pixels in the right view.

in which case the y-values in the disparity map would need to be used in the calculation of pixel correspondences, otherwise these are ignored. The x-disparity, shown in figure 3-9, is essential for the colour matching process since there will always be a difference in horizontal position of corresponding pixels in a stereo pair of images. The more accurate the data in the disparity map is, the better the end result will be and the fewer erroneous areas will be output, since an accurate disparity map means a high number of correctly identified correspondences between views. If a particularly noisy disparity map is being used then some blurring may be performed before passing it through to our algorithm – this helps to minimise the errors caused by bad matches.

After the algorithm has been performed on the input images, the colour difference map is blurred to remove harsh lines between grid sections. The resulting map (adjusted to show contrast better) can be seen in figure 3-10. The colour-adjusted image is created by multiplying each pixel's value in the colour difference map by the same pixel in the original image to be adjusted, for every colour channel. The results of this having been performed on the right-view image are shown in figure 3-11. These results can be better viewed with the checkerboard representation introduced earlier, and figures 3-12 to 3-14 show the results of this algorithm being successfully used. Figures 3-15 and 3-16 show the cropped results of subtracting a gaussian blurred version of the right-view image from its original before and after colour matching, respectively, in order to show the noise pattern in the images. It can be seen that these images are very similar, showing that the noise in the image has not been altered by the functions we have performed.

Figure 3-10: Blurred colour difference map.

In order to further highlight this, figure 3-17 shows the result of subtracting figure 3-16 from figure 3-15.

We show in figure 3-18 the results of our algorithm compared to those provided by [Corrigan et al., 2010] as well as The Foundry's ColourMatcher plugin provided as part of the Ocula toolset[(1)]. These results were produced using the default values of each plugin with no manual finetuning, the ColourMatcher plugin was also attached to an occlusion handling node as this is necessary for it to function.

[(1)] www.thefoundry.co.uk/products/ocula

Figure 3-11: Colour corrected right view image as output by our algorithm.

Figure 3-12: Once the images have been run through our algorithm, the colours are much more closely matched. This image shows the original left view in alternating squares with the colour corrected right view.

Figure 3-13: The colours match so closely that only by looking at the lamppost on the right is it clear that this image shows alternating squares from the original left and colour corrected right view.

Figure 3-14: The road portion of the scene also shows an extremely close colour match when viewed as a checkerboard with alternating squares of the original left view and the colour corrected right view.

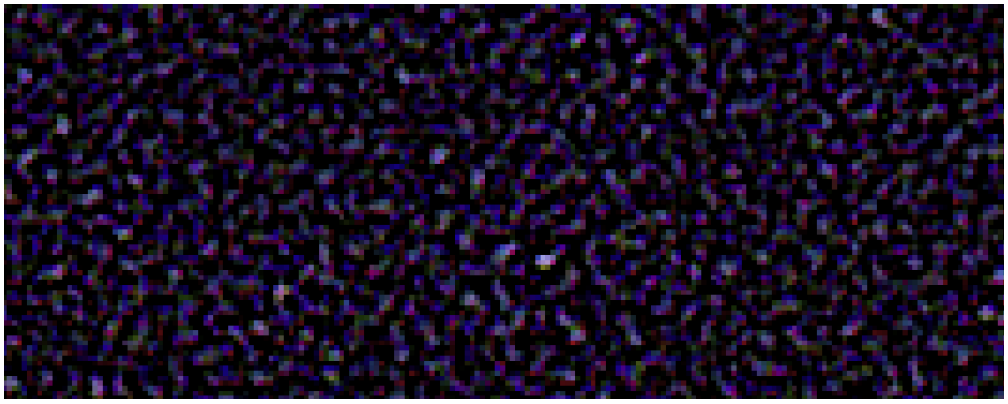

Figure 3-15: The noise pattern in the right-view image before colour matching.

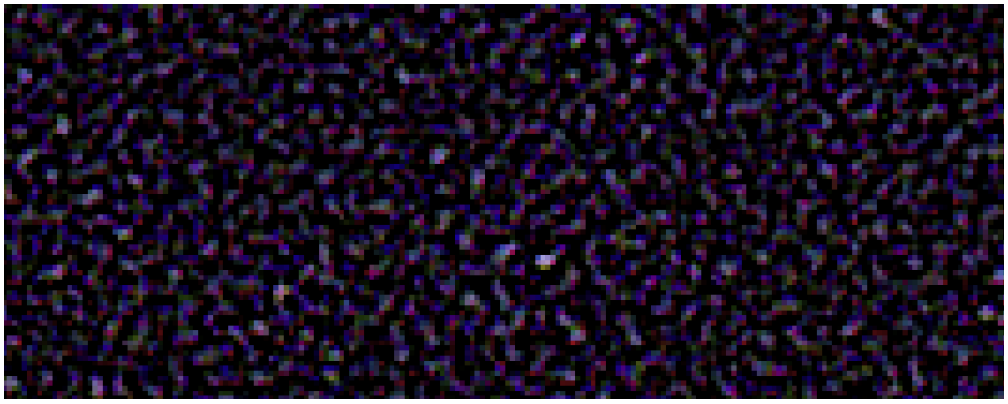

Figure 3-16: The noise pattern in the right-view image after colour matching.

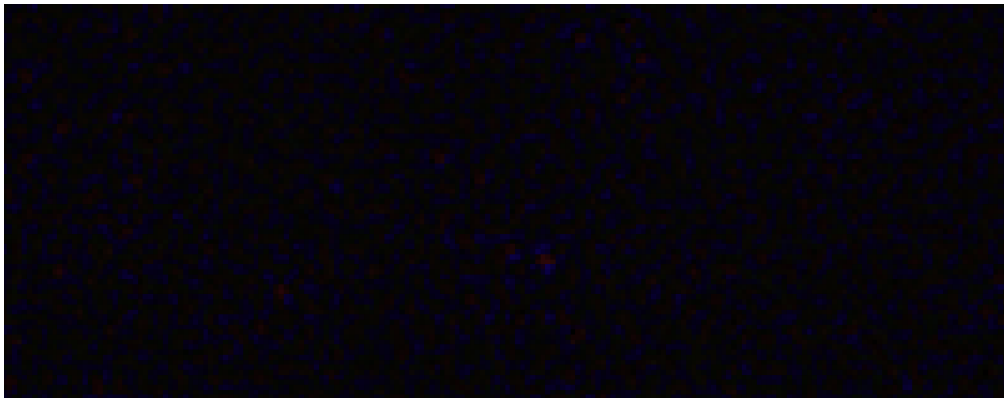

Figure 3-17: The difference between the noise patterns shown in figures 3-15 and 3-16.

Figure 3-18: Images from the Sigmedia Stereo Video Database [Corrigan et al., 2010] showing a) the original left view, b) the original right view, c) the colour corrected left view provided by [Corrigan et al., 2010] (using a linear Monge-Kantorovitch method [Pitié and Kokaram, 2007]), d) the colour corrected left view created using The Foundry's ColourMatcher plugin, e) the colour corrected left view using our EyeMatch method.

Figure 3-19: Image showing the results of colour correcting the right image using EyeMatch (top) and ColourMatcher (bottom). It can be seen that there is much less of a blurring effect when using our EyeMatch tool.

3.7 Discussion

The first technique that we tested was to simply calculate the average colour of all of the pixels in each image and blanket alter the target image so that the average would match the reference. This means that if the reference image was, on average, 1.2x more blue than the target, then every single pixel in the target image would have its blue value multiplied by 1.2. Unsurprisingly, perhaps, this results in incorrect matching in various areas of the image. This is because of the "difference between differences" throughout the stereo pair of images. Altering the right-view input image with an average value produces the result shown in figure 3-20, which may seem reasonable until viewed in our checkerboard representation for comparing views. Figure 3-21 shows the original left view in alternating grid squares with the right view which has been altered with the above method. It can be seen that, while achieving a fairly good match in some areas, there are clear colour mismatches elsewhere, such as in the sky covered portion of the images, making this a non-viable option.

We also tried histogram matching. Our algorithm would simply calculate the histogram of colours in one image, and alter that of the second image to match it as much as possible. This method requires pixel values to be placed into "bins" in order to calculate the range and frequency of colours appearing in an image.

In our tests of the histogram matching technique we tried both a global and a more localised approach. The global approach offers a good match to the tone of the reference image, and is very good for bringing wildly mismatching pictures more in line with each other, but on close inspection of our stereo test images there were just too many localised mismatches, as shown in figure 3-22. At this point, the idea of gridding the images was tested and the histogram matching algorithm was used on each grid section as if it were a separate image. This technique yields a better match of colours between individual sections of a stereo pair, but then gridding becomes noticeable in the final output. However, we found that there was no way of removing this gridding artifact without compromising the fidelity of the image produced.

It can be seen that our final colour matching method produces very good results for the majority of the image shown in figure 3-11. There are some limitations to the current method, which are discussed in section 3.8 but the amount of work required from an artist to overcome these limitations is minimal in comparison to the time cost of performing the entire colour match manually.

Figure 3-20: Results obtained with the use of an average colour difference for a global colour change.

Furthermore, we show in figure 3-18 the results of our EyeMatch plugin as compared to the ColourMatcher node included in The Foundry's Ocula toolset for Nuke. It can be seen that the results of both methods are superior to the global approach used in the Sigmedia Stereo Video Database example. Where we believe our method shows the most benefit can be best seen in figure 3-19. Here we see that the ColourMatcher approach transfers more than just colour information and in fact has a blurring effect on this image due to the fact that the left image is more blurred than the right in the sample footage. Our method, however, maintains the fidelity of the original image and only affects the colour.

Figure 3-21: Alternating squares of original left view, and right view which has undergone a global colour change based on average colour difference.

Figure 3-22: Alternating squares of original left view, and right view which has undergone a global colour change based on histogram matching.

3.8 Limitations

The solution presented is very effective in many cases, but it is not perfect. This section outlines some situations where our colour matching algorithm is not the best tool to use.

The main area where improvement would be desired is areas of occlusion, and figures 3-23 and 3-24 demonstrate why. It can be seen in figure 3-23 that the car has occluded the wall to such a degree that the lack of colour information from the other view has led to an incorrect match. This is largely due to the occlusion causing problems for the disparity map generator that we use, and is something that must be studied further in order to correct. Figure 3-24 shows further problems related to occlusion, which have been included here to demonstrate just how bad the problem can become, although this particular image shows the special case discussed in section 3.2 of this chapter and can therefore be disregarded.

It can be seen in figure 3-24 that the colour matching algorithm has completely broken down at the extreme side of the image (as expected), but although we can disregard this area due to the fact that it will be removed anyway it is important to note the manner in which the algorithm has failed. It is essentially an extreme example of figure 3-23 wherein the colour matching algorithm has found no appropriate corresponding area to find alternative colour information and so has simply taken this information from an inappropriate area. It is proposed that the better way of dealing with this particular problem would be to provide the algorithm with the extra information of which areas are occluded in the other view to allow for a special-case solution.

Another limitation is apparent when using this method on video sequences. The method of blurring each colour change map with each adjacent frame has one apparent failure case where the difference between each frame is too great and so not only is "flickering" still present, but the colours in each frame are incorrect due to being altered so much by the blurring. This is not a common occurrence, however, and so we can leave this to the artists to fix as and when the problem arises. As future work we would like to explore ways to eliminate this problem entirely.

Figure 3-23: Zoomed in view of a wall which is occluded by the car in the other view, causing an incorrect colour match.

Figure 3-24: Zoomed in view of the right hand side of the image which has no corresponding area in the other view – an extreme example of the error case shown in figure 3-23.

3.9 Conclusion

In this chapter we have presented a novel approach to the colour matching problem. By introducing a gridding process, we are able to maintain local details and not produce a result that is too general. This tiling of the images also allows for optimisation by way of threading and there is no reason these processes could not be moved over to the GPU to further improve speed. By removing outliers this algorithm is made robust to noise, and the method of ordering pixels before calculating the change required also means that the colour changes will be smoother and more subtle because the minimum change for each tile is always found and applied. A simpler solution, which used whole-tile colour changes, would require less processing and memory, but would also become more susceptible to tiling artifacts which would not be fixable with the current blurring method. There may be a case for offering the option of a simplified match, in order to obtain faster results and get a feel for the final output before running at the highest resolution, but with the optimisations of threading, and porting to GPU this will likely be a redundant feature.

The key contribution of the work shown in this chapter is the local accuracy of colour changes for stereo pairs. There are many methods of colour transfer, as discussed by [Neundorf et al., 2012], but none of them offer a perfect solution to the stereo problem. We are not suggesting that the work we have presented offers a *perfect* solution, but it is certainly an improvement on other approaches. The crucial difference between a colour transfer algorithm and a stereo image colour transfer algorithm is that simply matching the colour palette, or globally reshaping the colour histograms to match is not sufficient. As previously discussed, this sort of matching would result in some areas having a very poor local match.

There is a place for these global colour matching algorithms, however, since we have yet to deal with occluded areas. The two main ideas for handling occlusions, once they have been identified, are to either get the necessary colour change information from the surrounding area in the same frame, or to try and identify the exact same area in another frame and base the colour change on that. It may be sufficient, however, to perform a global colour match for the scene and apply the calculated changes to the areas lacking in their own specific colour difference information. While it is necessary to perform some level of colour change to an occluded area in order to blend it in with the rest of an image that has had its colour altered, there is a fairly large margin for error since there is nothing in the other view for it to be compared against. Based on this conclusion, it is clear that the best method for colour matching an occluded area will be to extrapolate the nearby areas for which an accurate colour difference has been found.

The problem with the idea of taking the information from another frame is that there is always the possibility that, for example, the lighting will change and so the colour difference altered and this area ending up with an incorrect colour change that is just as conspicuous as if it were left as is. Including surrounding frames in the algorithm will also use more memory and take more time for execution.

3.10 Future work

In order to overcome the limitations mentioned in section 3.8 we have identified several areas for future work.

Firstly, we assume in our approach that each colour channel is statistically independent and therefore deal with each one separately. It is likely, however, that a multivariate guassian distribution could be used in order to generalise and simplify the problem (using the Mahalanobis distance for our current noise threshold calculations). Given the results we are already achieving, however, we do not expect this investigation to yield particularly high rewards.

We also intend to investigate the effect of introducing a test on each grid section sampled to ascertain whether or not the colours are gaussian distributed as assumed. By performing a Kolmogorov-Smirnov test, or measuring the Bhattacharyya distance within a grid section we would be able to determine whether our subsequent filtering will be effective. This would allow us to either skip this section completely since an accurate colour difference will not be found, or to find some alternative means of dealing with the problem. One potential solution would be to use this information to create adaptive windows, splitting a non-gaussian distributed section into two or more gaussian distributed sections. This will be possible in many cases, such as when a strong edge is present in the section. The difference between a section of the expected colour distribution and that of one with a strong edge is shown in figure 3-25.

Finally, we can utilise the information in consecutive frames when calculating the colour difference in a given section. This should provide better results than the current method of calculating each frame in isolation and then performing a temporal blur between the colour difference maps.

Figure 3-25: Histograms of an area containing a strong edge (highlighted red) and of an area with fairly uniform colour (highlighted green).

Chapter 4

Vertical alignment of stereo images

In this chapter we discuss the importance of vertical alignment when dealing with stereo images, as well as exploring our methods of achieving it. This is a very common problem faced when dealing with stereo images, and is one worth solving considering how harmful any degree of misalignment can be to the viewing experience [Tam et al., 2011].

In appendix A, we describe how the epipolar geometry of a stereo pair of images can be calculated. Using this geometry, it is possible to rectify these images and thus guarantee a pure vertical alignment by ensuring all epipolar lines are horizontal. While this is a very powerful tool, with many uses in image processing, it is inappropriate for our purposes. The reason it is inappropriate here is because the severity of rectification could be such that too much of the original image would be lost once it is trimmed down to a usable cinema format. In addition to this, we are constrained by the need to maintain the original alignment of the "hero" view - the director will usually designate one camera in a shot as the "hero" and require the other view's properties to be matched to it. A further constraint we are given is to only shift areas by whole pixels as this tool will largely be used early in the pipeline to facilitate further processing and so there can be no blurring as this would make it more difficult to perform tasks such as matching tracker markers - a largely manual process.

We make use of the information in the disparity map provided as an input to our algorithm, and discuss the positive and negative aspects of relying so heavily on it. We offer multiple options to the user for correction of the vertical alignment problem due to the fact that there is no one technique that is appropriate for all situations. We developed a plugin for existing

software used in the Double Negative production pipeline, which is discussed in further detail in section 6.3. There are other tools available for vertical alignment, such as The Foundry's O_VerticalAligner[1] node, which implements many methods of vertical alignment including rotation and perspective warp. However, they do not implement the "group by x-disparity" method which we believe to be a very effective method of grouped alignment in many cases. We have implemented all of the methods discussed in this chapter to the point that they are useful in the production pipeline, but we still identify yet further improvements, as well as alternative methods, that may be implemented in the future.

The alignments we wish to produce are primarily for creating intermediate images within the workflow that give the artist something to work from. In order for these images to be as useful as possible to the artist, they must remain fully intact (i.e. with no blurring). In light of this, we will not be able to utilise shearing, scaling, or rotational transformations, and will need to round translation to the nearest integer.

4.1 Importance of alignment

There has been much research conducted in the area of stereoscopy, from a computer vision viewpoint as well as other areas [Backus et al., 1999], for a very long time [Wheatstone, 1838]. There has also been a lot of research conducted specifically into the discomfort and visual fatigue that can be caused by viewing stereo [Shibata et al., 2011], [Yang et al., 2012]. One very interesting point to arise through studying the literature from various fields, is that it is in fact possible for humans to fuse stereo images with a significant vertical disparity [Stevenson and Schor, 1997]. Cyclovergence allows the eyes to cope with a certain level of vertical disparity, but the continued presence of this disparity will result in eye strain and fatigue [Banks et al., 2012].

While there are several things that can cause discomfort when viewing stereo images, such as differing colours in each image or too great a horizontal disparity between views, the main thing we will look at here is vertical disparity. If the images are not correctly aligned vertically then the viewer will struggle to fuse the image and will suffer from fatigue and headaches in a very short time [Emoto et al., 2005]. We would like to create the best possible experience for the viewer, so it is imperative that we don't overlook the discomfort that can be caused by this issue. Even if we were to ignore the issue of comfort for the viewer, if the vertical disparity is so great that

[1] help.thefoundry.co.uk/nuke/9.0/content/reference_guide/ocula_nodes/o_verticalaligner.html

the audience are unable to fuse the images then the stereo effect is lost, and the film is unwatchable in 3D.

Figure 4-1 shows the anaglyph image of the original left and right view overlaid. Looking at this image through some anaglyph glasses is possible without discomfort and at first glance there may appear to be no problems with it. However, figure 4-2 showing the sign in the top right of this scene displays a noticeable vertical misalignment which will cause problems on a larger screen. Another example of this issue, which shows that it is a problem throughout the image, is present in figure 4-3. It is important to remember that while this appears as a small problem in these images, on a large screen (such as in a cinema) it becomes very noticeable. If you imagine that you are viewing a full screen version of figure 4-1, and that screen has a height of approximately 30cm, when looking at the erroneous areas there will be only a difference of perhaps 2mm. The BFI IMAX in London, however, has a screen height of approximately 20m[(2)], which would show the difference between views as over 13cm. Even accounting for the increased distance between viewer and screen, this is a huge difference that will result in the image becoming unviewable due to the fact that too great an angle will be subtended at the eye.

Nielsen and Poggio [1984] and Prazdny [1985] suggest that a viewer can still fuse a stereo pair of images with a vertical disparity of up to 10 minutes of arc, whereas Stevenson and Schor [1997] claims that this figure is more like 30 minutes of arc. If we look at the case of the IMAX theatre and assume a viewing distance of 12m metres (seating at the BFI IMAX is situated roughly between 12 and 24 metres from the screen), then the 13cm disparity previously mentioned would result in an angle of 37.24 minutes of arc being subtended at the eye. To reduce this to less than 30arcmin we would require the vertical disparity to be no more than 10.47cm and to reduce it below 10arcmin (to be on the safe side) we would need to keep the vertical disparity below 3.49cm. The experiments conducted in these papers, however, use only static images and do not account for the added strain of viewing many such images over an extended period of time.

While it is often possible to fuse these images when viewing them one at a time, even with a larger difference in vertical alignment than shown here, viewing a sequence will quickly become a painful experience. This is because of the added time required for the brain to make sense of and fuse the images, along with the fact that that time is simply not available when viewing 24 frames per second.

If the cinema screen mentioned above is used to present an 8k resolution

[(2)]http://www.bfi.org.uk/about-bfi/help-faq/bfi-imax

Figure 4-1: Anaglyph image showing left and right views overlaid.

film at the maximum height, then the 20m height of the screen is occupied by 4320 pixels. This equates to 2.16 pixels per centimetre. We would like to keep the vertical disparity found on these cinema screens to less than 3.49cm, and ideally much less. If we aim to keep at the lower end in order to ensure maximum comfort, then this still leaves us with up to a 7 pixel margin for error when dealing with full 8k resolution images. As the resolution decreases, however, we must shrink our margin accordingly and keep to within a few pixels in order to maintain a comfortable viewing experience for cinema-goers. Generally, we aim to keep to a smaller error (less than 1px) anyway, but this information at least allows us to feel comfortable in allowing a slight vertical disparity in order to maintain image fidelity – i.e. only shifting images by whole pixels wherever possible in order to avoid any blurring.

Figure 4-2: Anaglyph image showing vertical misalignment in top right of full image.

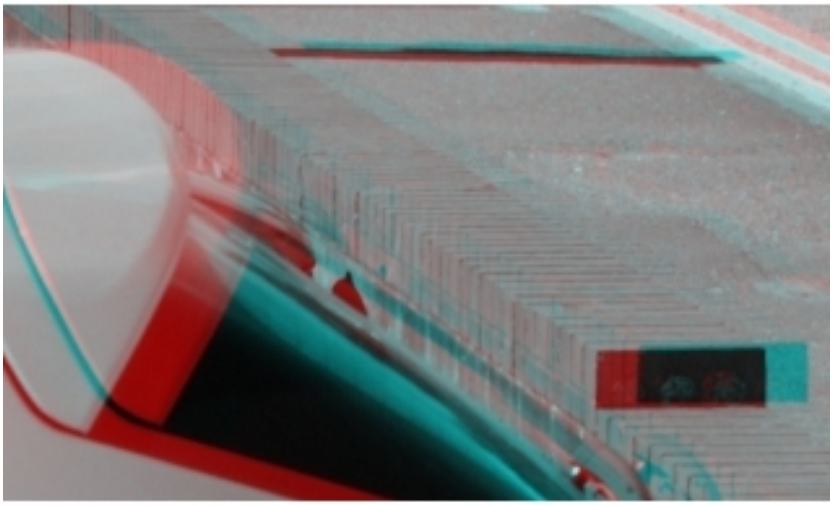

Figure 4-3: Anaglyph image showing vertical misalignment in bottom left of full image.

4.2 Problem description

Vertical misalignment of stereo images is common, and not at all easy to avoid. If the camera rig is static, as well as the lenses, but there is a slight error in setting up the cameras, then there will be a slight, consistent error on every frame of the shot. This is a relatively simple problem to solve since it only requires one adjustment which can be applied to every frame. Once that has been calculated it is just a case of verifying the adjustment parameters to ensure every frame aligns properly. If the camera rig is being moved during a shot, or if a zoom is performed, then the problem becomes more complicated. This can result in consecutive frames having slight differences in their misalignment, and therefore each frame must be calculated separately.

In addition to the static and dynamic camera rig problems, there are also two types of misalignment that must be corrected. Firstly, the more simple problem is a vertical misalignment that is consistent across the whole image. That is to say, that the problem lies in one of the cameras being slightly lower than the other, but pointing in exactly the same direction. This problem will require one of the images to be translated on the y-axis by some degree in order to bring it into alignment with the other. The more complicated problem is where rotation is involved. This problem results in different vertical disparities across one image because parts of the camera are moving in different ways relative to the scene. As a simple example of this, the left-most side may require a *downshift* of two pixels, while the right-most side needs to be moved *up* by two pixels. In this instance a simple y-translation will not suffice and a more sophisticated approach is needed.

The contraints we are given for this problem are as follows:

- The "hero" view must be respected - the other view must be matched to it without it being altered itself.
- Blurring should not be introduced by the alignment process, since the results will be used by the artist for further processing.

These constraints mean we cannot use classical methods such as image rectification in order to find an alignment, but must look at alternatives.

4.3 The contribution of this work

We present a collection of techniques for performing vertical alignment on stereo images. The main contribution of this work is our technique of grouping areas by their x-disparity before correcting their y-disparity, as we have not seen this method offered in other existing tools. The industrial contribution is in the form of a Nuke plugin which provides the four methods detailed below, making it easy for artists at Double Negative to utilise these techniques in their existing workflow (section 6.3).

4.4 Methods

We want to affect the fidelity of the original image as little as possible, since this tool will be used early in the pipeline with further work by the artist being applied. With this in mind it was decided that several options must be made available to the artists so that they can make decisions based on the requirements of a particular scene and sometimes even use different options for different parts of a shot. Sometimes it is necessary to adjust an image in sub-pixel accuracy, but this has a blurring effect on the image so where possible we would want to avoid this. We provide an option to take the values found in the disparity map and use them as they are, which creates blur. Alternatively they can be rounded to the nearest integer so that each pixel is moved in whole-pixel increments, this has no blurring effect but does sometimes result in insufficient matches. There are four slightly different methods of vertical alignment that were implemented in order to provide the necessary functionality to artists, these are described in detail below.

4.4.1 VerticalAlignment options

Figure 4-4 shows the y-values of a disparity map to be used as an input to the VerticalAlignment Nuke node in order to demonstrate the output produced by each method shown below.

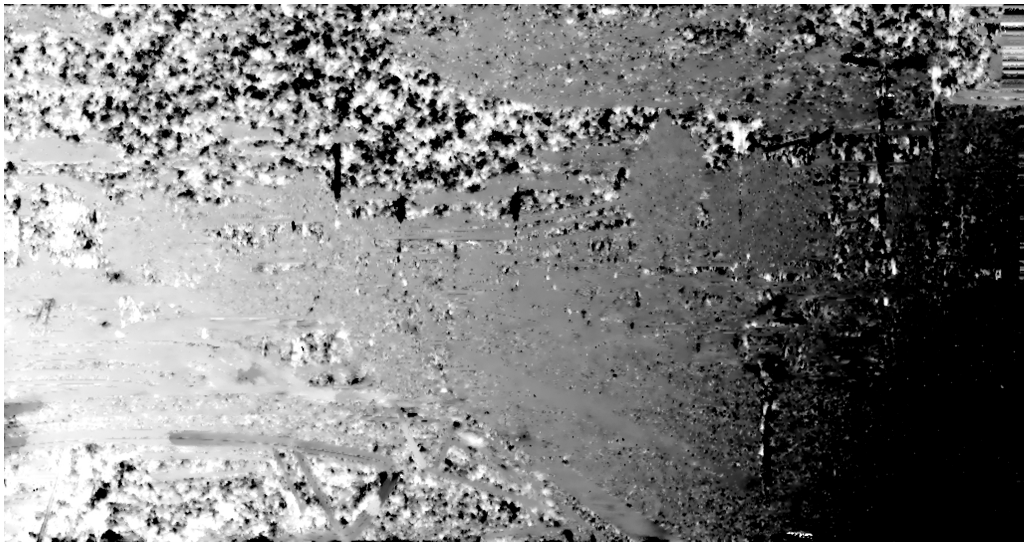

Figure 4-4: The y-values of the disparity map used as input, the colours have been adjusted to highlight contrast. Lighter colours represent a larger positive disparity, black shows a large negative disparity value.

1. Blocks

Areas close to one another are likely to have similar depth values, and so will require a similar correction for vertical alignment. With this option, the disparity map is first divided into grid squares with dimensions specified by the user. A mean-average of the y-disparity values within each grid square is then calculated and every pixel within this square is moved according to this average value. This process creates consistent shifts within grid squares, as shown in figure 4-5, and is useful when there are large areas that require the same vertical shift, but where there is a big variety in y-disparity values across the image as a whole. A good choice of grid dimensions will result in a good level of smoothing of disparity values and will virtually eliminate the problem of noise in a disparity map. This method is rarely effective for an entire image, however, as it is susceptible to tiling artefacts as shown in figure 4-6. These tiling artefacts will generally occur at the boundaries between objects that lie at different depths in the image. It is possible to make use of this method as a first step and then give further attention to these boundaries as an additional step in the process, however other methods are generally favoured by the artists dealing with these images.

$$\frac{\sum_{i=0,j=0}^{W,H} y_{i,j}}{WH} \tag{4.1}$$

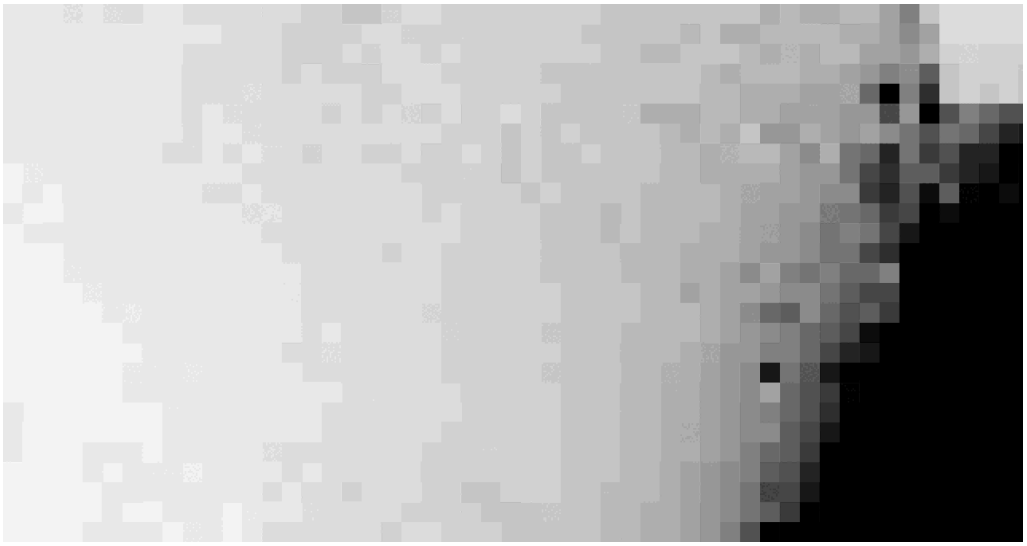

Figure 4-5: Result of grouping the disparity map into blocks before averaging y-disparity. The colour values have been adjusted to highlight contrast, with white showing a large positive disparity and black showing a large negative disparity.

Where i, j represent coordinates within the given block, W is the width of the block and H is the height.

2. Average

This method will do the same as the "blocks" method, but over the entire image. All y-disparity values are used to calculate the mean-average of the entire disparity map, and then the whole image is shifted as one. This can be useful for very flat images where the cameras are perfectly rotationally aligned, and there is very little variation in y-disparity. This is not often the case with stereo images, however, and this method will rarely be used on an image as a whole. It may, however, be used in conjunction with a mask to limit the area to a small part of the whole image. This is a fairly simple process, and since rotoscoping is almost always necessary at some point in the pipeline anyway, these masks can easily be used for the additional purpose of vertical alignment.

The equation for this method is the same as (4.1) except W and H become the width and height of the entire image, respectively.

Figure 4-6: Highlighting the potential tiling artefacts that can be created with the "blocks" method.

3. Group by x-disparity

This method works on the assumption that if areas have similar x-disparity then they are likely to be at a similar depth and so require a similar vertical shift. After we first look at the x-disparity of all of the pixels in the disparity map, we group pixels based on these values and a mean-average of y-disparity is calculated for each group. This is a very effective way of calculating consistent vertical shifts for several areas of an image at once, as can be seen in figure 4-7. This method can also be used in conjunction with masks to restrict the number of pixels included in the calculation and ensure that, in addition to having similar x-disparity values, pixels are also located in a similar region of the image. Due to the fact that masks are so easy to apply, and that separate areas of an image will often be dealt with separately anyway, it was decided that a "proximity check" within the algorithm itself was not necessary at this stage and would only serve to slow the algorithm down. In algorithm 1 we show the process of a simple bin sort, used to group together our pixels of similar x-disparity. By assessing the x-disparity (x) of each pixel (P) across the image (of width W and height H), we are able to sort these pixels into bins (or buckets) where the number of bins (n) has been defined by the user.

With all of the pixels sorted into the appropriate bins based on their x-disparity, we are then able to calculate the average y-disparity of each bin to apply to each area in turn.

A true depth estimate would require calculations involving the y-disparity, as well as other measurements such as the focal length of the cameras and

Algorithm 1 Group by x-disparity

```
1: bins ← new array of n empty lists
2: x_min ← x_0
3: x_max ← x_0
4: for i = 0 to WH do
5:     if x_i < x_min then
6:         x_min ← x_i
7:     if x_i > x_max then
8:         x_max ← x_i
9: interval = (x_max − x_min)/n
10: for i = 0 to WH do
11:     B ← floor((x_i − x_min)/interval)
12:     insert P_i into bins[B]
```

their baseline, but since we do not necessarily have this information and do not need to accurately calculate the true depth, we find that an estimate based on the x-disparity alone is sufficient for the purposes of grouping.

4. Blur y disparity

This is the only method which does not calculate an average of a group of pixels. It performs a Gaussian blur over the disparity map (or area within a specified mask) and shifts pixels by the resulting y-disparity values. This is effective in reducing any problems with noise within a disparity map, but once again will likely not be sufficient for calculating the vertical shift required across the whole image.

4.4.2 Closing remarks

The human visual system is very adept at dealing with various disparity issues, but is more sensitive to misalignments in the y-disparity than x-disparity (as detailed in section 4.1). As long as the x-disparity is kept within tolerable limits, we do not need to make any corrections to it at this stage. The y-disparity, however, is a bigger problem as the artist must be able to fuse the image in order to effectively work on it. By solving the y-disparity problem between the stereo images we solve the fusing problem, allowing the artist to continue their work. Unfortunately, due to the constraints imposed on us we are unable to use the classical methods of correcting this issue and so must take a more unconventional approach.

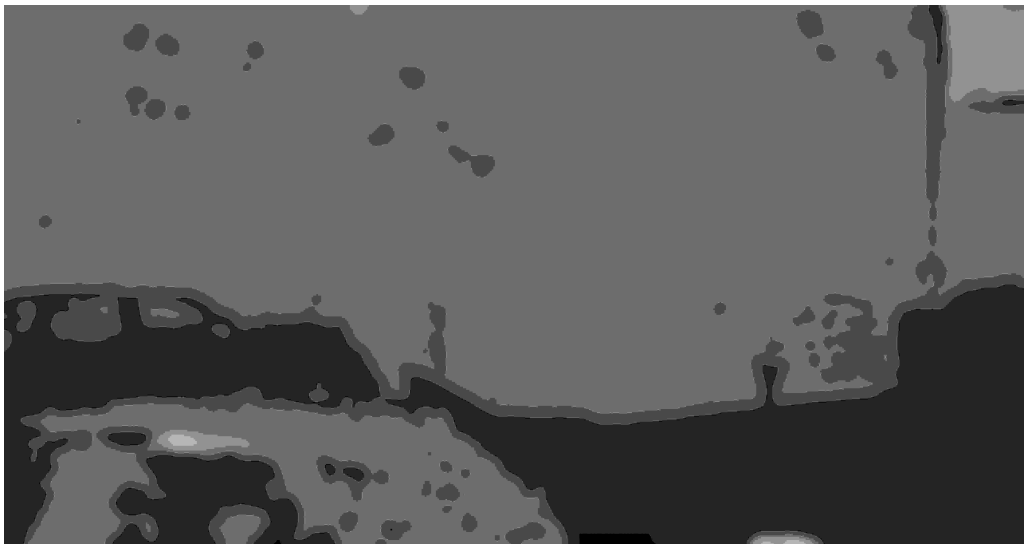

Figure 4-7: Result of grouping by x-disparity values before averaging y-disparity within each group. The colours have been adjusted to highlight contrast, with white showing a large positive disparity and black showing a large negative disparity.

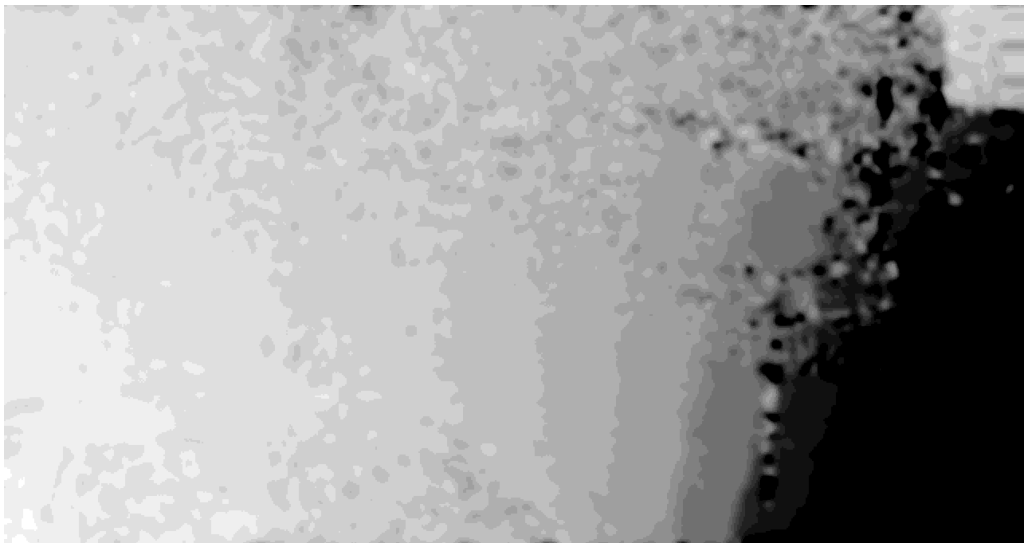

Figure 4-8: Result of simply blurring the original y-values of the disparity map shown in figure 4-4.

By changing the y-disparity locally, by integer values, we are essentially changing the position of the object in 3D space. As a result of this we are presenting an image to the viewer which is a distortion of the real 3D object, which is formally a bad thing but sometimes necessary for movies (and is the very definition of a "visual effect").

We state that we do not wish to alter the x-disparity due to the fact that any disparity changes will be an artistic decision further down the pipeline. Of course, by shifting the y position of objects in one of the views we are in fact altering the x-disparity (since $a^2 + b^2 = c^2$ in a triangle where the original disparity was c but now we are making b as close to 0 as possible and not altering a). This is a problem, but it is less of a problem than blurring would be. We could go so far as to calculate the x-disparity change necessary in integer values in order to keep it as close to the original value as possible, but ultimately this was decided to be unnecessary since the x-disparity can be fixed at the later stage. We are only interested in allowing the artist to fuse the images so that effects can be applied as needed.

A combination of methods is often required to vertically align images, hence the inclusion of many different options in our plugin. We have also provided the ability to do sub-pixel shifting within our developed tool as this is still sometimes necessary, though generally avoided. It is important to provide both of these options to give the artist as much control as possible. It is understood and accepted that there will be further edits required to make these results ready for presentation, as this tool is intended to be used for alignments in the middle of the pipeline which are useful to the artist rather than being ideal for the viewer.

4.5 Results and discussion

The success of our methods can be seen in the following images. Figure 4-9 shows the anaglyph image of both views overlaid after the left view has been aligned to the right with the use of the "blurring y-disparity" method described in section 4.4. The images are much better aligned and so will be more comfortable to view together. The images in figures 4-10 and 4-11 show cropped and zoomed versions of the same image where the improvement over figures 4-2 and 4-3 are clearly shown.

There are some limitations to this particular method that are highlighted in these images. In particular it can be seen in the area of the bottom left of the sign in figure 4-10 that the alignment has not quite been successful. In this case this is due to the fact that the disparity map contained errors

and a lot of noise, and so it is impossible to get a perfect result with this method.

Figure 4-12 shows the result of using the "average" method and contains none of the irregularities highlighted in figures 4-10 and 4-11. However, due to the fact that the vertical disparity between these images is caused by a rotation of one of the cameras, this method results in many parts of the image remaining misaligned.

The "group by x-disparity" method is best for avoiding the error shown in 4-10 since this creates an average of all of the pixels that make up the sign, minimising the effect of the noise causing that particular error. Figure 4-14 shows the results of using this method. Although this can achieve good quality results, it is the most labour-intensive of the methods since it requires an artist to specify how many bins they would like to use for grouping the x-disparities. This can be a difficult process, and contain some trial and error, since it is necessary to specify the correct number of bins for any particular image. Too many bins can result in some areas being over-segmented and there not being enough pixels to form a representative y-disparity, too few will result in large areas being vertically shifted by the same value when they should have been split into smaller groups which require different levels of correction.

The "blocks" method can also be effective in reducing this type of error, however it comes with the limitation highlighted in figure 4-6. Figure 4-13 shows the result created by this technique, and it can be seen that large areas of the image are satisfactory. The tiling artefacts present can be corrected manually by an artist and might still in many cases be less time consuming than the fully manual approach, however for an image such as the one used in our example the other methods are preferable.

Since the disparity map plays an integral part in all of the methods shown, errors in this will mean that none of the methods will be able to produce a satisfactory result. We find that some manual effort is required in these circumstances to fix the errors remaining.

The Foundry's O_VerticalAligner tool provides an option for using a perspective transform, but this was deemed too destructive to implement in our node since it does not maintain parallelism of lines. We have included the "average" and "blur y-disparity" options in our node for completeness, but it is possible to perform these tasks fairly easily with existing Nuke nodes. Our "blocks" implementation, however, reduces the amount of work required to achieve similar results and provides a useful tool for artists. We consider the main contribution of this work to be the "group by x-disparity" method, which can be very effective at making alignment corrections that

Figure 4-9: Anaglyph image showing aligned left and right views overlaid after using the "blur y-disparity" method to shift the left view.

vary across an image. This option is not provided by The Foundry and would be quite a lot of work for an artist to achieve manually. In section 4.6 we have identified several areas for further improvement of this technique to be implemented as future work.

Figure 4-10: Result of alignment using the "blur y disparity" option on top right of image.

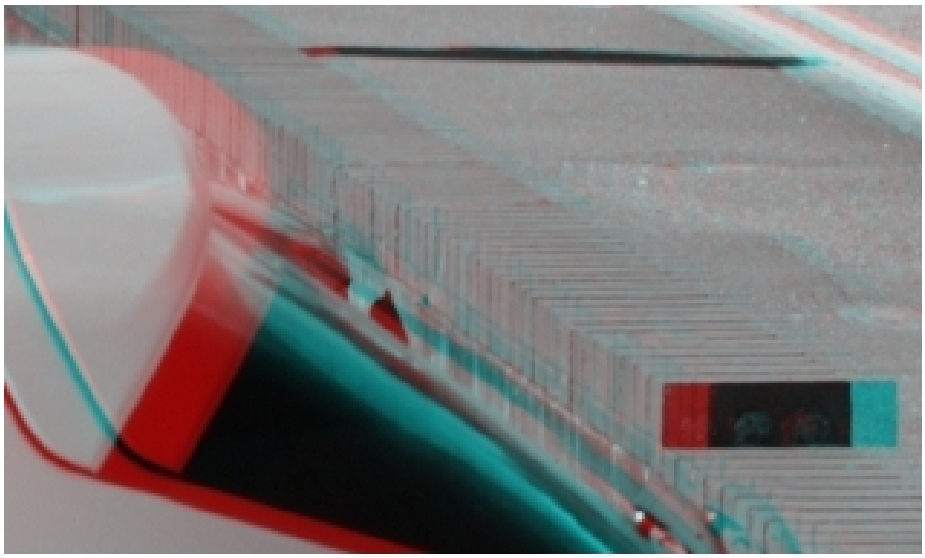

Figure 4-11: Result of alignment using the "blur y disparity" option on bottom area of image.

Figure 4-12: Anaglyph image showing aligned left and right views overlaid using the "average" method to shift the left view.

Figure 4-13: Anaglyph image showing aligned left and right views overlaid using the "blocks" method to shift the left view.

Figure 4-14: Anaglyph image showing aligned left and right views overlaid using the "group by x-disparity" method to shift the left view.

4.6 Limitations and future work

The "group by x-disparity" method currently has some limitations in the amount of effort that an artist must put in to selecting an appropriate number of bins, as well as potentially needing to perform the alignment of some areas of the image separately. An immediate improvement that could be implemented in the future would be to include some proximity checking to ensure that only contiguous areas of similar x-disparity are included in any group. This could extend to fully segmenting one of the images and then creating a separate bin for each segment, however this represents significant development time that we were unable to invest during the course of this particular project. To solve the bin selection problem we might further consult some of the segmentation literature in section 2.4. Another improvement that could be included with the "group by x-disparity" method would be to allow a gradient of disparity values from one side of a group to the other. This is necessary because of the case where a group covers a large proportion of the image from left to right and the misalignment is rotation based. This would have limited use, however, since maintaining fidelity is often a very important factor when using this tool.

Finally, it would be useful to artists if erroneous sections of the alignment were highlighted automatically to ease the identification and correction of such areas. This is a feature that is of much interest to Double Negative, not just for the alignment problem but also for generated disparity maps.

4.7 Conclusion

In this chapter we have presented an exploration of some of the techniques available for correcting vertical misalignment in stereo pairs of images, and have discussed the importance of correctly aligning images in order to provide a comfortable viewing experience for movie-goers as well as to make it possible for artists to actually work on the images. After investigation of the many possible methods for correcting vertical misalignment, and having identified advantages and disadvantages to each, we have implemented a variety of them. There are still further techniques that could be implemented but this was allocated a low priority since it became clear during this work that the best improvement to vertical alignment would come from an improvement in the disparity map that we use.

The main benefit of using the disparity map so heavily for our vertical alignment algorithms is that the bulk of the calculations have already been

done for us. It takes some time to create high quality disparity maps, but this research dissertation contains a variety of different tools that can all use it making it a worthwhile investment in processing time. The downside to this is that if the disparity map contains errors, then this will propagate through the pipeline and good results will never be reliably achievable. All of the methods described in this chapter concentrate heavily on mitigating the potential errors that can be present in a disparity map. Three of the techniques use some form of averaging, while one simply blurs in order to lessen the effect of mismatched pixels. If we were to use some other method, such as the affine transform or image rectification, then we might be able to achieve better alignment results but there is no way to do this while maintaining the full fidelity of each image.

Given the fact that an improvement in the disparity map was concluded to be the one thing that would provide the biggest gain for the colour matching algorithm in chapter 3, it is clear that this is an area that requires attention. The following chapter explores disparity map generation and our work towards developing a new algorithm utilising everything we know about the stereo footage being used at Double Negative.

Chapter 5

Using edge information and estimated epipolar geometry for improved disparity map generation

In this chapter we discuss a new approach to generating disparity maps between stereo images, based on estimated epipolar geometry and common edges detected in the scene. We believe this could provide improved results to tools currently in use at Double Negative. Our algorithm simplifies the problem by first using a Canny edge detector [Canny, 1986] on each image and using the result as a guide for calculating the disparity between key areas of the scene. We also utilise the SURF [Bay et al., 2006] algorithm as a means of finding appropriate feature points in each image that can be matched and used to estimate the relative camera positions and, therefore, the epipolar geometry [Zhang, 1998]. The aim of the work discussed in this chapter is to make a contribution towards the creation of cleaner disparity maps with crisp, clear edges.

5.1 Motivation for this work

Disparity maps are very useful in various parts of the VFX pipeline, and this solution will not only be used for the colour matching and vertical alignment algorithms discussed in chapters 3 and 4. For example, a good disparity map could be very useful during the rotoscoping process for stereo footage. Rotoscoping is a process which involves outlining specific objects in a scene

to create masks so that these images can then be manipulated. For example, if you would like to move an object from one part of a scene to another, you would first need to draw a mask around that object (rotoscoping) and then remove it and place it somewhere else. When dealing with stereo images, this process essentially has to be done twice. With a perfectly accurate disparity map it would be possible to rotoscope an object in one view, and then automatically perform the same operation on the same object in the other view. We do not set our sights quite so high as to try producing a *perfect* disparity map, but with sufficient accuracy we could at least save artists some time and allow them to use this technique so that they need just correct minor errors afterwards. There is also scope to use the corrections performed by the artists during this process in order to correct areas of the disparity map itself – this is explored further in the future work section of this chapter (section 5.8).

5.2 The problem to be solved

The algorithms discussed in chapters 3 and 4 rely heavily on an accurate disparity map being provided as an input. Throughout the course of development and testing, it was realised that the disparity maps that were being generated using Double Negative's in-house tools, and The Foundry's VectorGenerator node, could be improved greatly. Aside from containing a lot of noise, which in itself is a problem, a lot of disparity maps contain fuzzy edges and no handling of occlusions, such as that shown in figure 3-9. These properties result in the colour transfer algorithm looking in the wrong place for a corresponding pixel and so producing erroneous results.

The three main metrics for success for this algorithm are accuracy, robustness, and speed. There are existing solutions to compare against in order to gauge improvements, and an increase in any of these areas, without significant detriment to the others, will be deemed a success. Accuracy is the primary concern, since this project was started on the basis of the existing disparity map generators lacking in this area.

5.3 The contribution of this work

In this chapter we present our work towards a novel approach to disparity map generation and offer a contribution in both academia and industry. Many existing methods for disparity map generation are only tested on

relatively low resolution images, and until recently the Middlebury Dataset [Scharstein and Szeliski, 2003] contained only images up to a resolution of 1282x1110px which equates to just over 1.4 million pixels in each image. In 2014 a new set of images were added to the Middlebury Dataset which were of a higher resolution of up to 2880 x 1988px [Scharstein et al., 2014] but this still only equates to just over 5.5 million pixels per image. In the movie industry, we are regularly expected to handle images at the 4K resolution (3840x2160px, or just over 8 million pixels) and more and more frequently are having to deal with even higher resolutions, such as the 8K resolution which measures 7680x4320px adding up to over 33 million pixels per image. When faced with images of this size, many algorithms simply fail. We have made significant progress towards a disparity map generation algorithm which can cope with these large resolutions, is largely parallelisable, and which produces cleaner edges than existing techniques with a high level of robustness towards noise. We achieve all of these things by focusing largely on the prominent edges within the image and using a tiling system for much of the algorithm which allows many areas to be processed at once.

5.4 The algorithm

In this section we present our algorithm for disparity map generation, which is currently still in development. We separate this into two sections, here we present the elements that have been implemented and in section 5.5 we present the elements which are currently being implemented. We also provide extensive plans for future work in section 5.8.

5.4.1 SURF feature points

The first thing our algorithm does is use the OpenCV implementation of the SURF algorithm to find feature points present in each image. This algorithm is based on the Hessian matrix shown in equation (5.1).

$$\mathcal{H}(x, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{5.1}$$

where $\mathbf{x}$ is a point (x, y) in an image I at a scale σ, and $L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the Gaussian second order derivative. In practice, however, the computation is simplified by replacing gaussian filters with much simpler

approximations in the form of box filters. Full details for this algorithm are given by Bay et al. [2006].

We are very careful about the points that we select and will only use those with the very highest confidence values. We ensure an even spread throughout the images by making use of a gridding system. By first dividing the images into approximately 100 squares and then only selecting the best point from each square, we ensure that the points found are spread evenly over the entire image. This is done in order to avoid dense groups of points in a small area of the image, which would adversely affect the fundamental matrix calculation. Further, we then prune these results so that we are left with only the optimal points.

5.4.2 Random sample consensus pruning of feature match correspondences

In order to calculate the fundamental matrix, we need to pass the matching feature points in our stereo pair of images into the eight-point algorithm. We first prune the feature points by using Random Sample Consesus (RANSAC) [Fischler and Bolles, 1981] to detect and remove outliers from our dataset, since the eight-point algorithm is known to be sensitive to noise. The equation shown in (5.2) demonstrates the benefit of using RANSAC over simply testing all samples.

$$1 - (1 - (1 - e)^s)^N = p \tag{5.2}$$

Where e is the probability that a point is an outlier, s is the number of points in a sample, N is the number of samples required, and p is the desired probability that we get a good sample. Even setting our desired probability to 0.99 will result in much fewer samples being required by the RANSAC algorithm to detect outliers than if we were to test every pair of samples individually.

The RANSAC algorithm is still an active area of research, with alternatives and extensions still being explored [Moisan et al., 2016], however we find the classical method to be suitable for our purposes.

5.4.3 Epipolar calculations

We calculate the epipolar geometry of our two views, as detailed in appendix A. Once this has been done we can use the epipolar lines to calculate where further matches should occur, but we go one step further by rectifying the images so that we know matches should be on the exact same line. A certain margin for error is allowed and we add a 5-line buffer above and below the window used for future match checks. While we could use this information to calculate disparity on a pixel-by-pixel basis using some optical flow algorithm, this could result in some noise where erroneous matches are found. We wish to ensure we have clean edge matches at the very least, and so we simplify the problem further.

5.4.4 Canny edge detector

The Canny edge detection algorithm is used on the left and right images, with a fairly high threshold to ensure only the boldest edges are found. This will typically offer outlines of objects and people, which is a good place to start for finding clear matches and creating a clean disparity map. As a first step to the edge detection, we convert our image to grayscale and filter out noise with a Gaussian kernel K of $size = 5$ such that:

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{5.3}$$

Following this process, the Sobel operator can be used as a preliminary edge detection using the filters shown in (5.4).

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \text{ and } G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{5.4}$$

This process provides all of the information necessary to calculate the magnitude (5.5) and gradient (5.6) of edges found within an image.

$$G = \sqrt{G_x^2 + G_y^2} \tag{5.5}$$

$$\theta = tan^{-1}\Big(\frac{G_y}{G_x}\Big) \tag{5.6}$$

The final stages of the Canny edge detection algorithm are to apply non-maximum suppression in order to produce thin lines that represent only the edges and not the surrounding pixels, and to apply a hysteresis threshold in order to only keep the strongest edges as well as "good enough" areas which are connected to them.

While looking for matches on a pixel-by-pixel basis would be prone to errors, finding matching edges is a much more successful venture simply because there are fewer areas that look similar.

5.4.5 Harris corner detection

We also locate corners with the Harris corner detector to provide another method of assessing the confidence value of matching areas. The Harris algorithm works by maximising E in equation (5.7) which measures changes of intensity I given a shift of window $w(x, y)$ by (u, v).

$$E(u, v) = \sum_x \sum_y w(x, y)[I(x + u, y + v) - I(x, y)]^2 \tag{5.7}$$

By performing a Taylor expansion on the function $f(x + u, y + v)$ we arrive at the first order approximation of $f(x, y) + uf_x(x, y) + vf_y(x, y)$, therefore we arrive at the approximation in equation (5.8).

$$\sum [I(x, y) + uI_x + vI_y - I(x, y)]^2 \tag{5.8}$$

Which becomes:

$$\sum u^2 I_x^2 + 2uvI_xI_y + v^2 I_y^2 \tag{5.9}$$

We can then rewrite this as a matrix equation to produce (5.10).

$$\sum \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{5.10}$$

So that we are finally able to say that:

$$E(u,v) \cong \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{5.11}$$

Where:

$$M = \sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{5.12}$$

Further details of the Canny and Harris algorithms are given by Canny [1986] and Harris and Stephens [1988], respectively.

5.4.6 Matching edges

In order to match the edges in the first instance, we simply use the information that we have already calculated along with some assumptions about our images. Firstly, since we have rectified the images, the match should be vertically aligned with our reference edge. However, we assume that there is a margin of error in our rectification calculations and so allow a buffer for each window checked. We match areas using a gridding process. By using fairly large squares (50x50px) we ensure that there is plenty of edge information in each window, but if we were to make them much bigger there would be less chance of finding a good match due to differing disparities of different edges. For each 50x50px window in the left image, L, we pass a 50x50px window over an area of 60xWpx in the right, R, where W is the full width of the image (since all we know about the matching area is that it should be *somewhere* along the line).

In order to calculate the best matching area, we simply start in the top left of the search area and calculate the number of matching edge and corner pixels between views. In order to find the best match for edge pixels, we seek to minimise the following equation:

$$\left| \sum_{y=0}^{50} \sum_{x=0}^{50} L(s+x, t+y) - R(u+x, v+y) \right| \tag{5.13}$$

where (s, t) = coordinates of bottom left of L, and
(u, v) = coordinates of bottom left of R, such that
$0 \leq u < W - 50$ and $t - 5 \leq v < t + 5$

This method essentially equates to calculating the Jaccard measure of our two windows, and selecting the right-hand image window which results in the highest value. We can say that A is the set of all edge pixels in our left image window, and B is the set of all edge pixels in our right image window such that we wish to maximise:

$$\frac{|A \cap B|}{|A \cup B|} \tag{5.14}$$

For the corners, we simply check to see if any corner pixels from the right window have coordinates within a radius of 1px from the coordinates in the left window. Each matching corner pixel improves the confidence of a match.

5.4.7 Refine edge matches

The edge matches must then be refined since the initial matches are rather crude and will not be much use on their own. The first thing that must be done is to check any areas that were ranked the highest of all options, but which are still vastly lower than their surrounding results. This will often be the result where the window contains two or more objects which are at very different depths and therefore have very different disparities. Sometimes, however, it is simply the case that the previous step has identified the wrong area as being the most likely match. Our first step in improving these areas is to look at the grid squares immediately surrounding them. As each result is assigned a confidence value based on the quality of the initial match, we can perform what we call a "confidence boost". This is a simple belief propagation which involves assigning a square with low confidence with the disparity values of an adjacent high-confidence square and assessing the quality of the result. If the result is comparable to the square's previous result then it is given the new disparity values. Algorithm 2 shows the basic process of comparing the confidence (c) of tile a to tile b. If tile b has a

greater confidence then we assess the confidence that a would have if given the same disparity values ($disp$) as b. If these values are sufficiently close, then we assign tile b's disparity to a.

Algorithm 2 Confidence boost

1: **if** $a_c < b_c$ **then**
2: $\quad a \leftarrow b_{disp}$
3: $\quad$ **if** $a(b_{disp})_c > 0.95a_c$ **then**
4: $\quad\quad a_{disp} \leftarrow b_{disp}$

5.4.8 Refine group matches

In order to correct areas where one square contains multiple objects at different depths, we must split the edge information into separate groups. Groups are defined as collections of adjacent pixels, and so an edge will usually create two distinct groups (but not always). By doing this, we are then able to perform the same checks as above, but with a smaller reference area. This means that objects at different depths *should* be tested separately and so a better match should be found. The groups that we split our edges into are found during the initial edge detection process where we expand the found edge to add thickness and then subtract the original edge. This gives us two groups for every edge, and proves to be very useful since it is often the case that significant depth changes will be found either side of an edge of some sort.

For the group matching process we use the monochrome version of the input images. This provides our matching process with significantly more information than just the existence of edges, allowing more detailed testing to be performed. At this stage we are still using the earlier technique of performing subtractions on the windows of each view in order to assess which disparity values are the best. We are still using integer values for the disparity here, with a view to refining later in the algorithm.

5.5 Currently being implemented

The above is a breakdown of the work that has been completed so far on this project. Due to the promising results that have been produced by the steps already implemented, active development is still ongoing. Below are the planned steps that will be developed in order to create a complete and

usable tool. In section 5.8 we discuss ideas for the further development of this tool beyond this initial algorithm.

5.5.1 Sub-pixel refinements

At this stage we no longer just work in whole pixels and start to calculate with sub-pixel accuracy (referring back to section 2.3), since we are trying to refine down to the most accurate disparity map possible. At this stage, since we now have the necessary image information again, we are able to check the areas that lie either side of the edges. This means that we can make use of some of the segmentation techniques found in section 2.4 in order to find larger areas of disparity information. As we have said before, the idea with this is to avoid per-pixel calculations as much as possible and so reduce noise in the final disparity map – as long as the edges are accurate, there should be a high accuracy in the areas in between. Problems will only arise if there are undetected edges that prove to be significant further along the pipeline.

5.5.2 Fill in gaps

Finally, it is necessary to "fill in the gaps" in our disparity map, since we only have the edges mapped at this point. To do this, we first look at the surrounding edges to identify our areas of interest (i.e. any area between two edges). We can then look at the disparity information that we have already found for the side of each edge that faces inwards to the area we are looking at. If the disparity values for these areas are the same (or at least very similar) then this is a good sign, because it means that we have likely found an area with a constant disparity. The next step is to look at the full colour version of this area and utilise some segmentation technique to ascertain if it belongs to the same object. If it does then we can simply assume that the whole area is at the same depth and can be given the same disparity value. If the edges have different disparity values but are deemed to be the same object, then this could be attributed to something like the side of a building which has one edge much closer to the camera than the other. For this we can in-fill with a gradient between the extreme values.

Figure 5-1: An example of the distribution of SURF feature points used for the rectification calculations, after pruning.

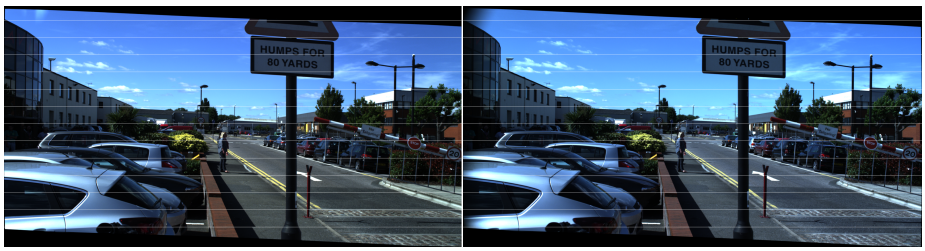

Figure 5-2: A pair of rectified stereo images with lines to show the accuracy of the rectification.

5.6 Results

In this section we outline the progress made towards producing accurate disparity maps that can be of real benefit in the post-production pipeline. This is highlighted for both our previous algorithms mentioned (chapters 3 and 4) and for other areas not covered in this document (such as stereo rotoscoping or matchmoving).

Due to the fact that the SURF results are pruned so heavily, and that a gridding method is used in order to ensure a greater distribution of feature matches than might otherwise be achieved, the rectification process produces some well aligned images. Figure 5-1 shows example SURF points, and 5-2 shows an example of an aligned stereo pair of images which are ready for the next steps of the epiFlow algorithm.

Once the high-threshold Canny edge detector has been run over the images, the next step is to perform a broad sweep of them in order to find the initial matches. Figure 5-3 shows the edge-only images, and figure 5-4 shows an example match found using only edges and corners. At this stage the matches are based on edges and corners (found with a Harris corner detector) only. Matches are assigned a confidence value based on the number of matching edge and corner pixels in the given window – this results in areas with no edge or corner information being given an initial confidence of zero.

Figure 5-3: Initial matches are found using these images which show only the existence of edges, all other image information is ignored at this stage.

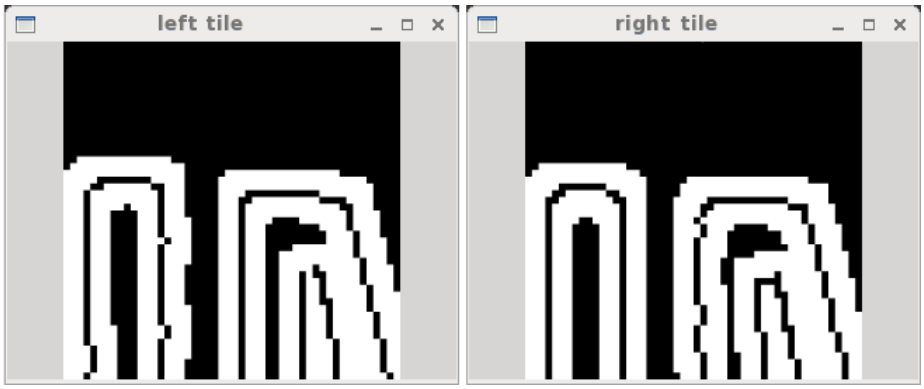

Figure 5-4: An example match found by comparing the difference between windows and selecting the lowest value candidate.

A group of initial matches is shown in figure 5-5.

After the initial matches have been found, the next step is to perform a "confidence boost". This entails looping through every window with the highest possible confidence value, and then checking the confidence value of all adjacent squares. If an adjacent square has a lower confidence value then the disparity value of the high-confidence square is applied to the low-confidence square and the "correctness" is compared with its original disparity value. If the correctness is comparable to the square's original value then its disparity is changed to the new value and its confidence is increased to match the central square. By performing this action, we eliminate most of the errors caused by there being several possible disparity values for a given square. Figure 5-6 shows the same area as figure 5-5 after this process has been performed.

Since each step so far has only been concerned with edges and corners, and has dealt with whole tiles, we find several erroneous areas at this stage. For example, figure 5-7 shows an example whereby a perfect match has been found for an edge, but it is not the correct edge.

Figure 5-5: Initial whole-window matches based solely on edges and corners. White squares in the right image show high confidence matches, with confidence values decreasing in the order red-green-blue-turquoise.

Figure 5-6: Whole-window matches after the "confidence boost" has been performed. White squares in the right image show high confidence matches, with confidence values decreasing in the order red-green-blue-turquoise.

Figure 5-7: The initial match found by the algorithm is not the correct edge.

Many of these cases are corrected by examining the monochrome image data of individual groups within each tile. A group is defined as any contiguous collection of pixels, and each tile may contain many of these groups. The case shown in figure 5-7 contains two groups, one for the edge of the foreground post and one for the cloud in the background. It can be seen in figure 5-8 that the result is much closer to what would be expected once this task has been performed. Due to the fact that the cloud is occluded by the post in the right-hand image, an accurate match is impossible. However, this is exactly the case where we believe occlusion detection should be a trivial matter since it is clear that one group matches very well and the other does not and so occlusion must be present.

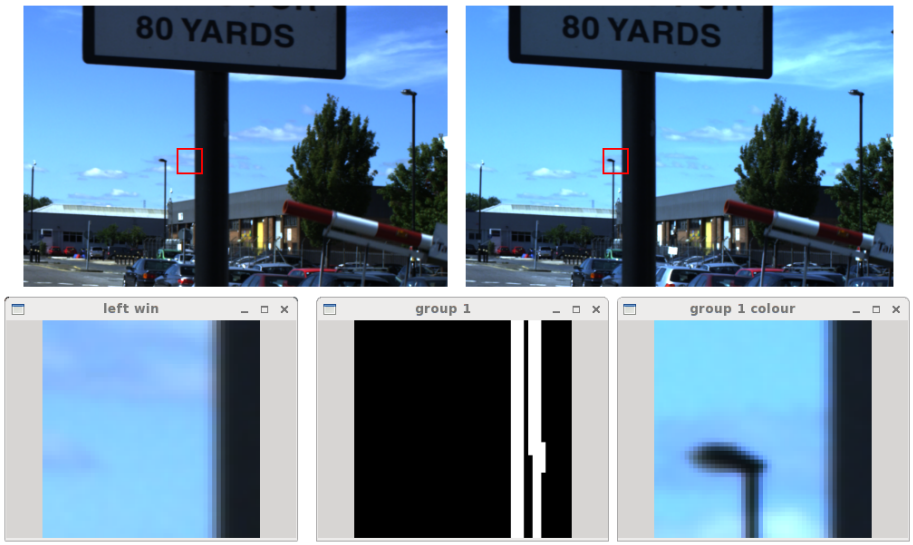

Figure 5-8: Using the image data, rather than just the edge information, at this stage allows the correct match to be found.

5.7 Conclusion

In this chapter we have presented a novel approach to the problem of disparity map generation. By constraining the problem in the many ways that dealing solely with stereo image pairs allows, we have created the foundation for a fast, robust, and accurate algorithm for producing disparity maps with a real usefulness in the industry. The work-in-progress results given in section 5.6 show the current strengths of the algorithm, although its efficacy will not be known until a complete disparity map can be created and compared with the results of the alternatives mentioned. Due to the fact that the interaxial distance is always very limited, and that the cameras are always horizontally aligned (imperfectly, but still close enough for our purposes), we are able to include certain constraints in our calculations and produce a higher quality result than if we were to try and create a more generalised solution for any two images taken from vastly different viewpoints.

Our aim with this project was to provide the foundations for an improved disparity map generation algorithm at Double Negative. In that regard, we have made very good progress. However, there is still a lot more development to be done in order to make this a usable tool for the working pipeline, which we discuss in section 5.8. We have shown that our technique is capable of finding accurate matches, but we have not yet produced a finished disparity map or implemented all of the planned refinements. The results that we have produced so far are sufficient to encourage further development

in this area and there are many tools that will benefit once work is complete. With research into optical flow techniques constantly pushing forward, it is easy to imagine that this project will continue to be developed and refined for many years as the techniques and results continue to improve. At the outset, we listed speed as a primary concern for the finished tool, but this will be addressed towards the end of development once the tool has already proved useful. At the early stage of usage it will be acceptable to request that artists set the job going overnight to use the output disparity maps the next day.

5.8 Future work

Although we have made a lot of progress with this research there is still a long way to go to make a tool that is useful in the production pipeline. This is very much an area of interest for Double Negative and so active development will continue due to the fact that the benefits of getting this right are enormous. In section 5.5 we have outlined the immediate development plan in order to complete the initial implementation of this tool, and in this section we will discuss the work that will follow.

One thing that we are keen to do as soon as we have a finished product that can produce the output that we need, is to compare it to the existing solutions that have been developed by others. In order to get an idea of where our algorithm stands in the ranks of other disparity map generation techniques, we will use the Middlebury Dataset images[(1)] and use their ranking system to see how epiFlow compares. As we have already discussed, the images available in the Middlebury Dataset are not quite of a high enough resolution to be fully representative of the images that our algorithm will frequently have to deal with, but they will still be useful in allowing us to assess the strengths and weaknesses of our algorithm compared to others.

We discussed in chapter 3 the need for some form of occlusion handling. Performing this operation at the disparity map generation stage [Sun et al., 2014a], rather than in the colour matching algorithm itself, allows the results to be utilised elsewhere in the pipeline as well. There are many uses for disparity maps, as previously discussed, but there are also many instances where an occlusion mask would also be helpful to an artist.

Due to the fact that epiFlow matches each side of a found edge separately, we can easily see if there is a large discrepancy between the locations of their respective matches. If there is a large discrepancy, then this is usually

[(1)] http://vision.middlebury.edu/stereo/data/2014/

Figure 5-9: Due to the vertical misalignment in the circled area, it is possible to deduce that some occlusion must have occurred in one of the views.

indicative of the presence of some occlusion. What we mean here is that if, for example, there are two areas which are examined which are next to each other in one view, but vertically misaligned in the other view, then this probably indicates that one object is being occluded by another and the match on that object has occurred at a different point. Figure 5-9 shows a potential scenario in which we can detect occlusion based on the fact that there is some vertical misalignment present.

There are essentially two possible types of occlusion that we need to detect, the first is outlined above. The second type is where the matches still occur as expected because the object being occluded has a fairly uniform texture, and so there is no way to differentiate between one part of it and another. An example of an area where this might occur can be seen in figure 5-10. To find these types of occlusion, we have several context cues to use as a guide. Firstly, these areas of uniform texture can be "attached" to areas containing features, and so the occlusion can be found through detecting any discrepancies between what is expected and what is found. Secondly, we can look at both of the edges of the area found and ascertain the size of the area in each image. If there is found to be a difference between views then we can often assume that some occlusion has occurred. Estimating the size and position of this occlusion is possible using the information we already have regarding it. The position of the occlusion can be ascertained from the surrounding objects and the knowledge of which view contains the occlusion, and the size of the occlusion can be estimated from the differences between visible area sizes.

In addition to occlusion detection and masking, we aim to provide the colour matching algorithm with alternative areas to find appropriate colour information when the actual corresponding area is occluded. This is a problem that requires further investigation since there are many possible options, and deciding which is best for any given situation may prove to be non-trivial. In some cases, the best area to find a colour match may simply be the area immediately adjacent to the occluded area. At other times this

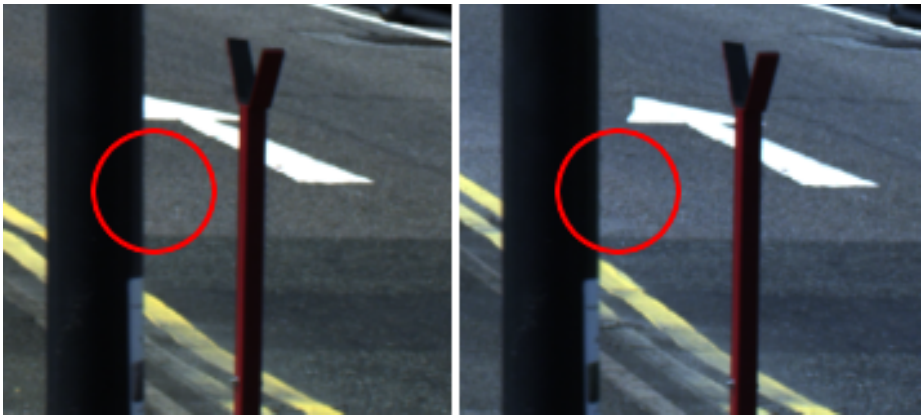

Figure 5-10: This type of occlusion can be difficult to detect due to the fact that a close match is still found for the disparity because the area visible is identical to the area occluded.

may not be suitable and the match may have to come from another frame, when the area is no longer occluded. Lighting changes throughout a scene may make this method impossible, however, and so the colour match may need to be taken from a completely different area which has a similar colour. This could provide its own set of problems, however.

Another useful addition to this will be the integration of a tool that has been developed separately at Double Negative to allow artists to create a smooth gradient by selecting corner points of an area. Using this image editing method on a noisy disparity map will allow users to manually clean up areas that require smoothing. However, we would also like to use the same method in the creation of the disparity map to provide smooth areas in the first instance. An example of where this will be useful is where we have the side of a building in shot which spans foreground and background. In this example, the foreground part of the building could have a very different disparity to the background part, yet because they are the same object the change in disparity will be smooth. Rather than calculate each pixel within the area separately and risk noise in the disparity map, we should achieve much better results by calculating the disparity of the foreground and background sections before filling in the area between with a smooth gradient from one to the other.

A further development in this area would be to allow an artist to interactively edit the disparity map to correct errors seen in the stereo images. For example, while rotoscoping a stereo pair of images, the artist could work on one view and have the mask automatically added to the other based on the information in the disparity map. The artist could then correct any errors manually and have the disparity map automatically adjusted based on these changes so that it is correct for future tasks.

Finally, 360° video and virtual reality experiences are a growing area of interest and producing live action footage for these is still very difficult. Expanding our epiFlow algorithm to work with multiple cameras in a 360° setup will enable many problems to be solved with far less manual input than is currently needed. For example, stitching all of the images together in the first place is a very labour-intensive task.

Chapter 6

Tools developed

In this chapter we discuss the practical tools which have been created throughout the course of this research. Although the academic contribution is important, it is also worth noting the significant contribution made to the industry. In chapters 3, 4, and 5 we described the various algorithms that have been developed but without going into too much detail about how we have made these solutions available to artists at Double Negative. Here we provide the details of the tools we have developed, either as plugins for The Foundry's Nuke compositor or as a standalone application.

6.1 What is Nuke?

Nuke is a node-based digital compositing application which is used by many visual effects companies, including Double Negative. It allows users to complete many of the tasks required in the post-production pipeline including, for example, lens distortion correction, retiming, and tracking. A node in Nuke is a plugin which performs a specific task. In the basic package there are, for example, nodes for blurring or colour grading. Nuke is used heavily at Double Negative, and so the decision was made early on that developing plugins would be the best way to turn research results into practical industrial contributions. By implementing our algorithms as nodes in a widely used software package, we allowed the artists to use the tools as they were being developed and received some invaluable feedback during the process. There were several artists who were willing to test out the beta-nodes before there was a project to work on, and so by the time there was a stereo show for them to be used on they were in a stable condition for reliable and effective use.

6.2 EyeMatch: a tool for colour matching between stereo pairs of images

EyeMatch was the first major Nuke node to be developed as part of this research, and was a direct progression of the work outlined in chapter 3.

6.2.1 Contribution of this work

The development of this Nuke node offers a clear industrial contribution as it provides a fast and effective way of colour matching stereo images, saving time for the artists involved and allowing them to work on other things.

EyeMatch has undergone thorough testing and improvement based on artist comments. Because of this, it has proved to be very useful in the post-production pipeline at Double Negative. Most of the testing was performed on shots that were being processed as part of a bid for work, and as such cannot be published, but recently (2014) the node was used extensively for the Ridley Scott film "Exodus: Gods and Kings". During the time that this movie was being worked on, a node-tracking app was implemented at Double Negative, offering the opportunity to view hard data regarding the use of the EyeMatch node. This app tracks the number of times a node is created by each user, but unfortunately will not register a new node if the original is simply copied, up-versioned or reused with different inputs. As a result of this, many users are recorded as only creating a single instance of the node when in reality they would have used it for many different shots. "saw" is my own username, so the high number shown for this user is simply on account of testing.

Figure 6-1 shows the different users who included the EyeMatch node in their Nuke scripts. In addition, figure 6-2 shows the usage of the node over time – the records only go back as far as August 2014 which was, unfortunately, towards the end of the Exodus project.

These charts offer clear evidence of an industrial contribution being made by the EyeMatch node.

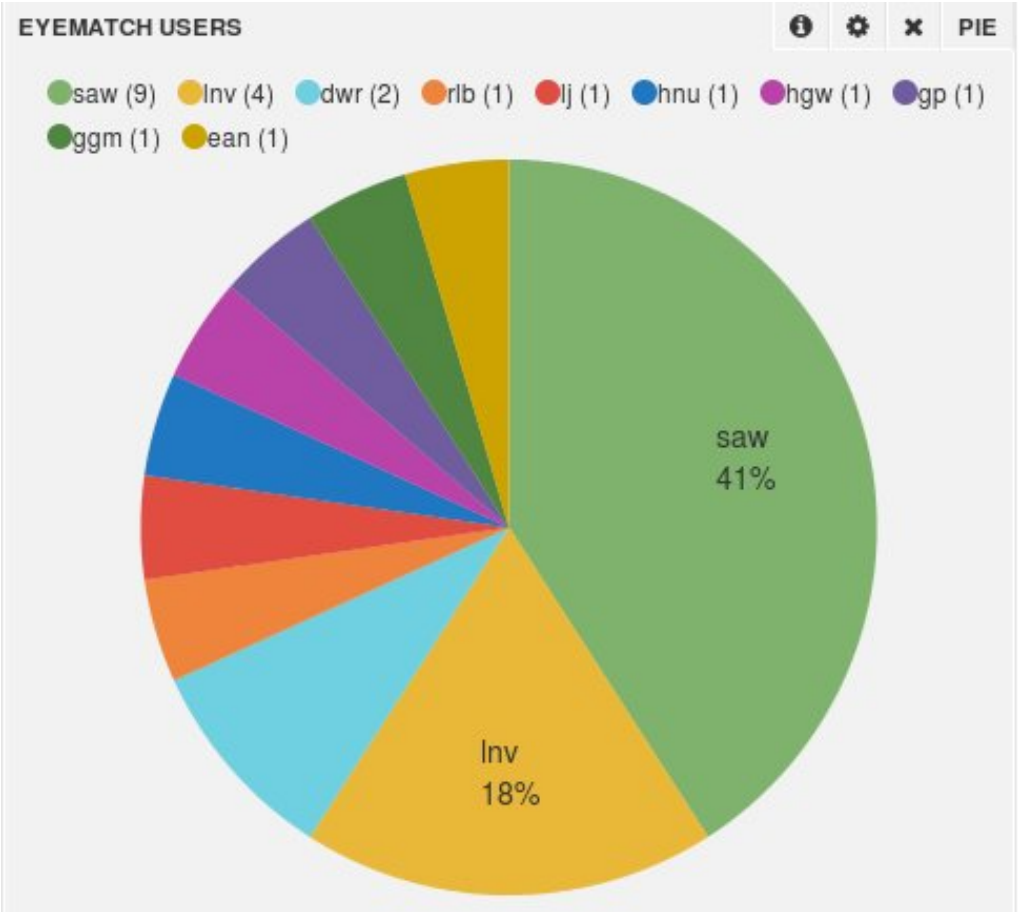

Figure 6-1: Pie chart showing the various users who created instances of the EyeMatch Nuke node.

Figure 6-2: Histogram of EyeMatch node instances created between 1st August and 1st November 2014.

6.2.2 Description of the EyeMatch node

This node takes as its input the left eye view, the right eye view, and the disparity map describing the relationship between the two. Figures 3-9 and 4-4 show the x and y disparity information for our example scene, while figures 3-4 and 3-5 show the input images themselves. A flowchart of the algorithm performed within the EyeMatch node is shown in figure 6-4, but a brief description is given here.

The first process is to blur the input disparity map, the degree of blur is controlled by user input. This allows an artist to compensate for a particularly noisy disparity map, or to simply soften some edges if they deem it necessary. If more advanced blurring techniques, or other processes, are required above and beyond the provided gaussian blur, then it is possible to apply these before attaching the disparity map to this node. It is to allow for that kind of workflow that we implemented this as a Nuke node.

We then use the disparity map to warp one of the images so that it matches the other as closely as possible, this saves us having to calculate where to find the corresponding pixels for any given area as they will now have the exact same coordinates in each image.

Everything is now fed into the EyeMatchInternal node, which is the name given to the C++ program which performs all the necessary calculations to create the colour difference map. An in-depth discussion of this can be found in chapter 3.

The output to the EyeMatchInternal node is a colour difference map describing the exact per-pixel changes required to match the colours of one view to the other. This colour difference map can be used to match the colours by multiplying each pixel with its corresponding pixel in the view to be adjusted.

Due to the fact that the image is gridded before processing, at this stage there will often be some very distinct lines at the borders of these grid squares. The simple solution to this is to blur across these boundaries in order to create a smoother result. From an artist's point of view, it is better to have a smooth result overall even if this comes at the expense of accuracy to some parts of the image. This is the final stage for a single pair of images, but for a sequence we must finish with some temporal smoothing to avoid flickering on playback. This is accomplished with a standard temporal median filter provided by Nuke which produces the results we are after.

The necessary image can now be multiplied with the colour difference map

using Nuke's Merge node and the colour match is complete.

6.2.3 User defined parameters

The user interface shown in figure 6-3 shows the parameters that can be specified by the user in order to adapt the algorithm for different situations, the features of which are described below. Once a user changes one of the parameters, it is applied immediately. Depending on the resolution of the images, the updated results will be displayed in the Nuke viewer in a matter of seconds.

Grid size.

The grid size can be specified, and does not need to be defined as square since height and width can be adjusted independently. This is important since there is always a tradeoff between desired smoothness and required precision. A smaller grid size will result in a greater precision (as long as a good quality disparity map is provided as input) but may produce a less robust result as a smaller grid square will mean fewer pixels being looked at when deciding what constitutes an outlier. This means there is more chance of some noise sneaking through. Conversely, a larger grid size will result in a smoother result but could produce some errors around edges, especially those with large occlusions.

Blur.

The user is also presented with two blur options. The first of these is for blurring the input disparity map, which can help when the disparity map is of a lower quality. This will have a negative impact on precision but is very helpful for smoothing out noise. Secondly, the user may specify the blur of the output. Some blurring is always necessary in order to lessen the obvious boundaries between grid squares, but sometimes more will be needed or less would be desired in order to get the necessary result. The user is also given the choice of a median filter, which is appropriate for most tasks, or a slightly more sophisticated, but much slower, bilateral filter.

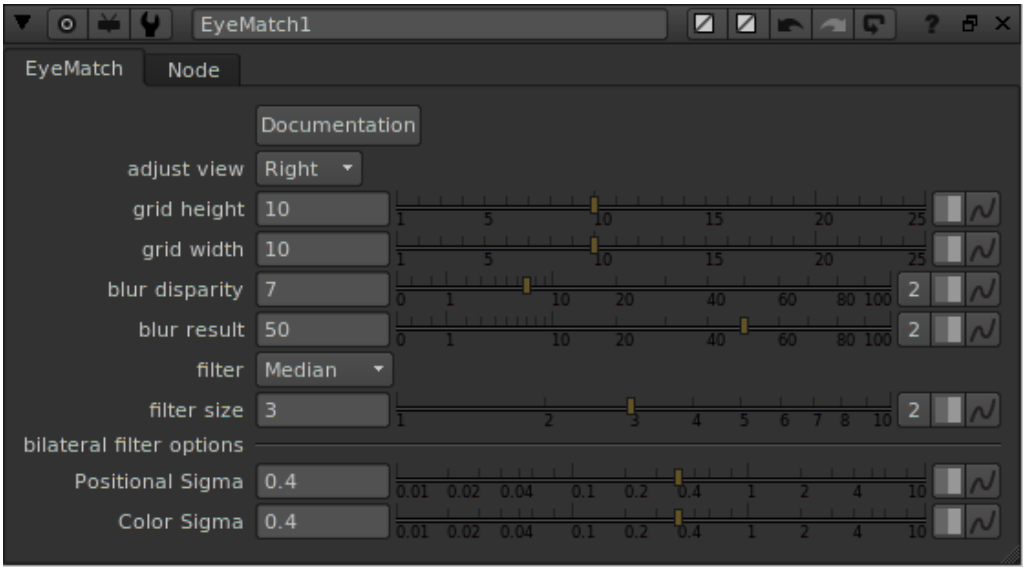

Figure 6-3: The user interface.

6.2.4 The algorithm

This section outlines the algorithm for the EyeMatch Nuke node. Figure 6-4 shows the basic workings of the EyeMatch node while figure 3-3 shows the inner working of EyeMatchInternal, an additional program written in C++ to perform tasks not well suited to the nodes provided in Nuke. The elements highlighted in red show functionality provided within Nuke.

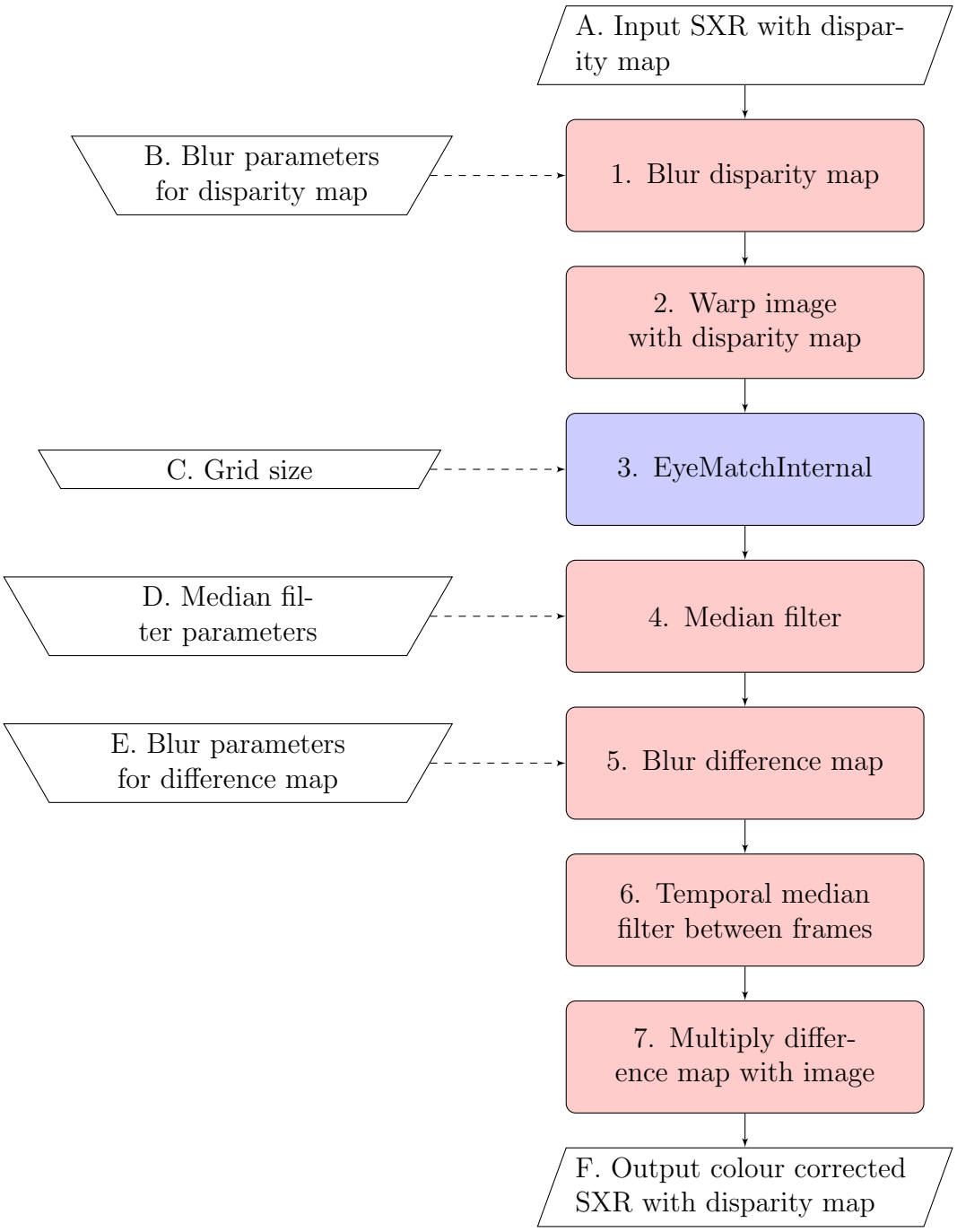

Figure 6-4: Flowchart showing the EyeMatch algorithm used for colour matching in Nuke

A The input to this algorithm is an SXR image with disparity channels. The SXR format is simply an extension of the EXR format for stereo images. The EXR format has been used at Double Negative for many years and so it is essential to be able to handle it.

B This allows the user to set the level of blurring for the input disparity map. The optical flow algorithm can produce disparity maps of varying quality and so sometimes it is necessary to blur more.

C The grid size is set by default to 5x5 as this provided the most reliably good results in testing but occasionally a larger or smaller grid is preferable so this was made an option to the user.

D Again, the filter size doesn't often need to be changed from the default but this was made possible for the occasions when it is necessary.

E The resulting difference map requires blurring to lessen the impact of the gridding process and may need to be altered depending on the size of grid used.

F The output will contain all of the same information as the input except one of the views will have adjusted colours to match the other.

1 Blurring the disparity is an easy way to lessen the effect of poorly matched pixels.

2 The iDistort node in Nuke allows an image to be warped on a per-pixel basis using the information in a disparity map. This is used to align one view with the other for future processing.

3 EyeMatchInternal is shown in detail in figure 3-3.

4 The median filter helps to remove erroneous pixels in the difference map by setting the centre pixel to the median value of all pixels within the window given.

5 A further blur of the difference map smooths out any remaining erroneous values to make them less apparent in the final image.

6 The TemporalMedian node in Nuke will look at a given pixel in the current frame as well as the one before and after and set its value to the median, this helps to lessen flickering over an image sequence.

6.3 VerticalAlignment: a Nuke plugin for correcting vertical misalignment

VerticalAlignment is a plugin developed for Nuke which implements all of the methods described in chapter 4. The idea behind creating a plugin for Nuke is that the artists at Double Negative are already familiar with the software and use it on a daily basis. By creating a plugin we bring our research directly into the pipeline and allow easy access for artists to make frequent use of it. The plugin takes as input an SXR (stereo EXR) image with disparity channels, it outputs an SXR with one adjusted view and disparity information that has been adjusted to represent those changes.

6.3.1 Contribution of this work

This plugin has undergone a similar development process to the EyeMatch node and has been tested by artists "in the field". "Exodus: Gods and Kings" is the only project that has been completed at Double Negative that could make use of this tool, since it is stereo-specific, and so much of the feedback for it has come from the team responsible for that movie. The contribution of this work is clear and it has found a place in the production pipeline and shown itself to be a worthwhile addition to the artists' toolkit. Vertical alignment can be a very time consuming and labour intensive task, this is the case with or without our node, but VerticalAlignment makes the task less time consuming in all cases and in most will significantly reduce the amount of effort required from the artist. In summary, the contribution of this node is to allow artists to perform the frequent task of alignment quicker as well as allowing them to perform further work which would not be possible with misaligned images.

6.3.2 Description of this node

The input to this node is a stereo image with disparity information (stored in additional channels). The node will change one of the views as well as some of the disparity information, which is necessary as we will be using it to change one of the actual images. If, for example, a pixel in the disparity map says there is a y-disparity of +4 pixels but due to the method used the pixel in question is actually moved up by 3 pixels, then that pixel in the disparity map must be moved up 3 pixels and have 3 subtracted from its value. The x-disparity channel will not be altered by any of the vertical

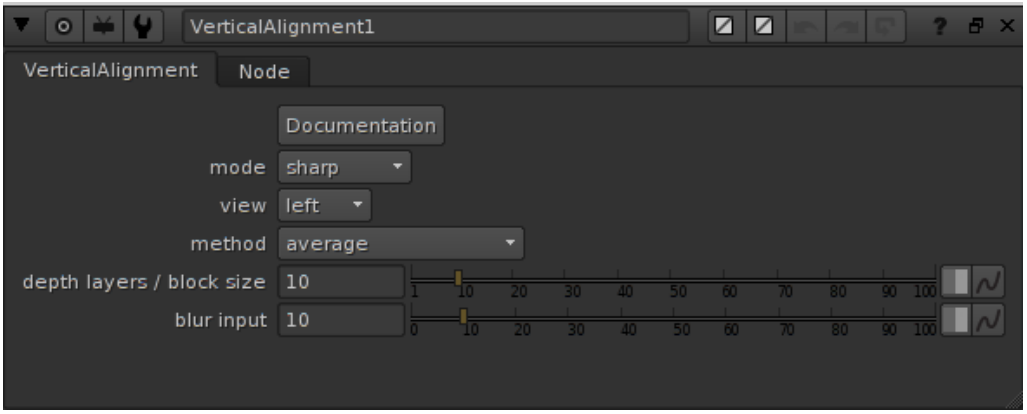

Figure 6-5: The user input options for our VerticalAlignment node.

alignment methods used, because we intentionally avoided any method that would do this as it could negatively impact the stereo effect the director was intending (or even completely ruin a scene by forcing the disparity past that which is comfortable for a viewer).

6.3.3 User defined parameters

The user interface for the VerticalAlignment node can be seen in figure 6-5. Under the "mode" menu, an artist can select either sharp or accurate, with sharp meaning that pixels are only shifted in integer increments and accurate meaning that they are shifted with floating point precision. The "view" option allows the user to specify whether it is the left or right view that is shifted in order to match with the other. "Method" contains the four options detailed in chapter 4, and the remaining two sections are parameters for these various options.

6.4 MagicWarper

This is a set of three tools: AccumulateMotionVectors, VectorAccumulator, and MagicWarper. These tools were developed together and are all used in conjunction in order to properly make use of the available motion vectors. By creating three separate tools we make it easier for the user to navigate the various steps and options that we have made available.

By accumulating motion vectors from any given frame, we are able to calculate the correct positioning for any paint added to our single reference

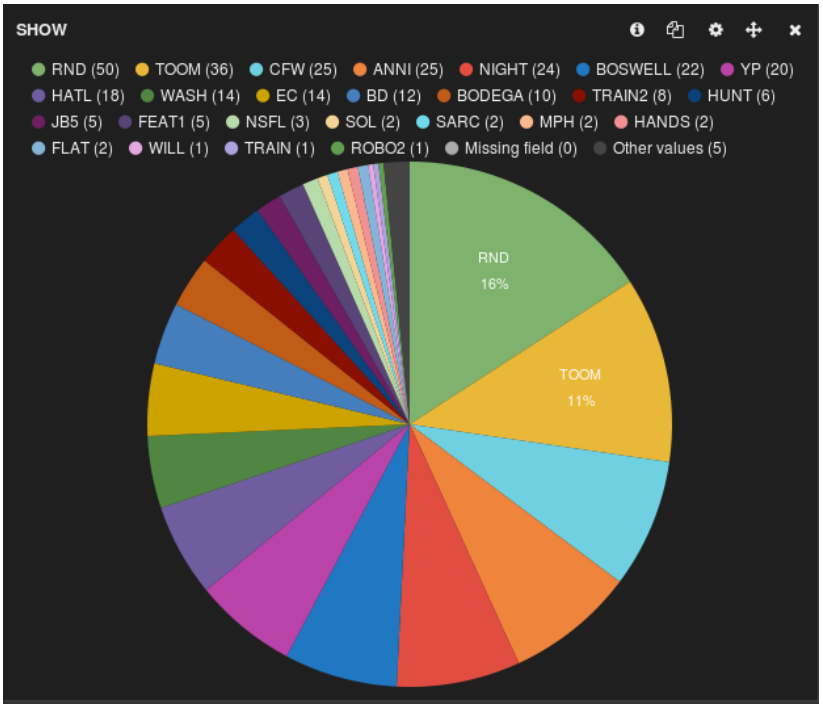

Figure 6-6: A selection of projects that have used the VectorAccumulator tool. These include Assassins Creed (TOOM), Fantastic Beasts and Where to Find Them (BOSWELL), and Emerald City (EC).

frame. In addition, we are able to use these accumulated motion vectors to perform a stabilisation, ensuring that every frame keeps a given area in the same position and countering any unwanted camera movement while allowing the rest of the scene to move as it should. Our approach allows for an arbitrary frame to be selected as the reference, and for any number of keyframes to be added.

6.4.1 Contribution of this work

These tools have had a huge impact at Double Negative and have been used on many big blockbuster movie and smaller TV projects. In figure 6-6 we show a chart of some of the projects that have used the VectorAccumulator tool, and in 6-7 we show those that have used MagicWarper. They continue to receive very positive feedback from artists and will continue to be developed as more creative uses for the tools are discovered.

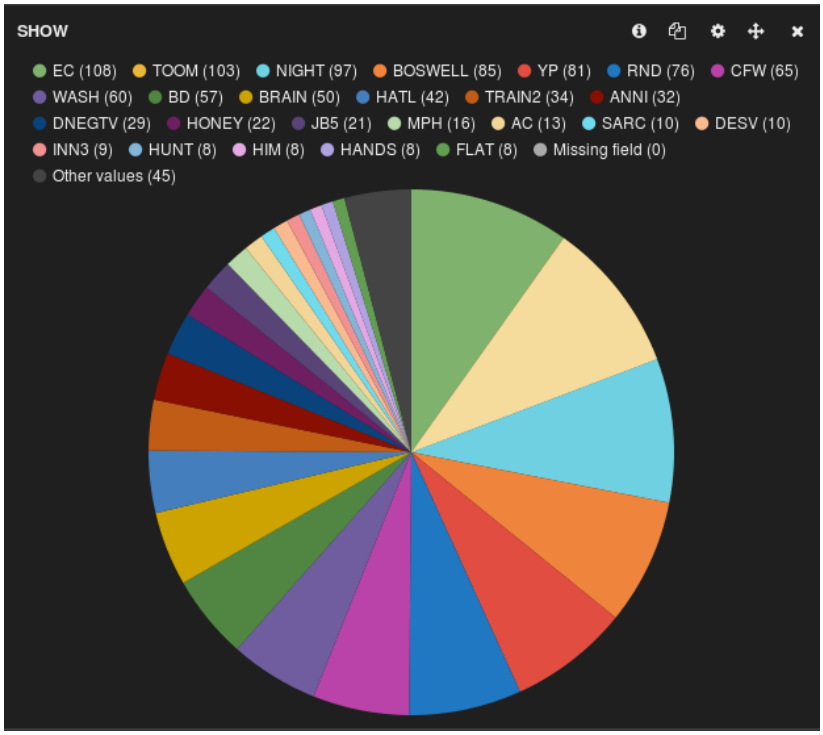

Figure 6-7: A selection of projects that have used the MagicWarper tool. These include Wonder Woman (NIGHT), The Young Pope (YP), Cure For Wellness (CFW), and Jason Bourne (JB5).

6.4.2 Description of this tool

In the movie post-production pipeline, there are often situations in which some element will need to be added to a scene. This could be anything from graffiti on a wall to dirt added to a moving vehicle, but throughout this section we tend to focus on digital make-up on an actor's face and body as this is one of the more difficult real-world examples we have come across so far. Often an artist will simply have to create many keyframes throughout a sequence in order to maintain the illusion that the digital make-up is in fact a part of the performance captured by the camera. We have created a tool that minimises the time spent on this laborious task by utilising motion vectors which are almost always created for each sequence anyway. In order to save time during the rendering process, and to allow an artist to produce results quicker when working on a shot, we create several layers of pre-accumulations which are used instead of individual frames. Although this adds a time-cost to the initial creation of the motion vectors, the amount of time it saves during the interactive use of our tool more than makes up for it. There is also a space-cost due to the fact that we are storing pre-accumulated vectors for bigger frame jumps than usual, but thanks to the development of our sparse algorithm, we have cut this space-cost down from ls to a maximum of $2s$, where l is the number of pre-accumulation layers and s is the space taken up by the original vectors. This use of a sparse dataset can slightly increase the time taken to output results, but not significantly enough to cause concern, especially when accounting for the huge space saving.

This tool was developed as a means of reducing the amount of time spent by an artist in applying effects that were to be present throughout a large part of a sequence. As a matter of course we regularly create motion vectors anyway, and so we wanted to make further use of these in order to track pixels through an entire sequence. Since we already have several methods of creating these vectors, we decided to focus on a tool that could simply use any pre-calculated vectors or even those calculated on the fly if necessary. The reason adding effects to a sequence is normally so time consuming is because the usual way of working is for an artist to manually create keyframes in order to define the movement of the added effects in such a way that they appear to be a natural part of the scene. If the motion within the sequence is not very smooth then this can result in a lot of keyframes being needed and therefore a lot of work is needed by the artist.

Accumulating motion vectors is, unfortunately, not as easy as simply adding together the motion value at each pixel with the same coordinates for every frame. We must first use the vectors of the next frame in order to find which pixel of the current frame corresponds to it, and then add those

values together. This is a very computationally expensive operation since we need to read every pixel of the next frame individually, and be able to randomly access any pixel from the current frame once we have calculated where the corresponding values lie. For a small number of frames this is not too difficult, but performing this operation across, say, 100 frames becomes very sluggish if performed on each frame individually.

We are using "pull" vectors, which tell you where to get the corresponding pixel *from* in the previous or next frame, rather than "push" vectors, which tell you where the current pixel will go *to* in the previous or next frame. The benefit of pull vectors is that you are guaranteed not to have any gaps in your resulting image, whereas with push vectors there can be many pixels that end up empty when a good correspondence is not found. It is for this reason that we opted for pull vectors, and therefore why we must talk about "backwards" vectors when we are moving forward through a sequence, and "forwards" vectors when moving backwards.

6.4.3 Pre-calculated motion vector accumulations

As previously mentioned, our MagicWarper tool takes motion vectors as an input and does not calculate these itself. This decision was taken because of the amount of time it takes to calculate strong motion vectors, which would result in a much less interactive tool, as well as the fact that we already have several ways of creating these vectors and may wish to develop further ones in the future, so there was little sense in including a specific method in the design of this tool. The accumulations that are created are only as good as the vectors that it is given, and so we would like to keep ourselves open to using better vectors in the future as techniques become faster and more accurate.

It was decided that the power-of-2 accumulations should be pre-calculated and stored on disk to facilitate the interactive use of this tool. Initially this involved storing a dense dataset of all of the accumulated vectors, meaning that for every frame n we would store the vectors to frame $n+1$, $n+2$, $n+4$ and so on, typically up to $n+32$. As this tool became more popular, however, it became necessary to develop a system that could use a sparse dataset in order to conserve disk space. Where the dense dataset up to layer $n+32$ would take up $5\times$ the space of the original vectors, the new sparse system allows an arbitrary upper limit (we calculate this based on the frame range of the sequence, so a 550 frame sequence will contain accumulated vectors up to $n+512$) and will only take up a maximum of $2\times$ the space required for the original vectors. We achieve this by only calculating each accumulation a at every a^{th} frame, resulting in some slightly

longer processing times but a much reduced impact on disk usage.

We build up the accumulations sequentially and recursively, using the same rendering template and simply feeding in the previous accumulation results in order to create the next. The process for calculating the accumulations is as follows:

Read in vectors

We take single-frame motion vectors as an input to our algorithm and so the first accumulation layer we will calculate will be the vectors for 2-frame motion. For this, we must read frame n and $n+1$ to calculate the 2-frame motion from $n-1$ to $n+1$. For each accumulation layer we calculate, we must read the f-layer motion vectors for frame n and $n+f$ where f is the previous accumulation layer.

Find corresponding vectors in each frame

The vectors in frame $n+f$ are used to find the corresponding vectors in frame n, the vectors in frame n can then be used to find the corresponding vectors in frame $n-f$. Because we are using backwards vectors, we must first read frame $n+f$ and work backwards towards $n-f$.

Accumulate the vectors

The original $n+f$ vectors are added to the corresponding vectors in frame n, this creates the necessary vectors to jump from $n-f$ to $n+f$.

Repeat as needed

The resulting vectors can then be fed back into the same template to create the next layer. For example, if we used the 1-frame vectors to create the 2-frame accumulations then we can now feed back in the 2-frame vectors to create the 4-frame accumulations.

Originally it was up to the user to specify how many layers they wanted to create and these accumulations would be calculated for every single frame. Now we have implemented the option to create a sparse dataset and the

upper limit is automatically calculated. To calculate the upper limit we simply round the frame range of the sequence down to the nearest power of 2 value. Our sparse dataset always begins at the first frame and will calculate all of the accumulations from that point. This means that in a 70 frame sequence, it is possible to jump immediately from the first frame to frames 2, 3, 5, 9, 17, 33, and 65. Where it would be possible to perform similar jumps from any frame with the dense dataset, we must now perform some best path calculations in order to utilise the sparse dataset. So, where it would be possible to get from frame 2 to frame 66 in a single jump with the dense dataset, we must now follow the slightly longer path of 2->1->65->66 with the sparse dataset. We are able to go backwards as well as forwards in our accumulation leaps because we have both forward and backward vectors available to us.

6.4.4 On-the-fly motion vector accumulations

The first step to performing the interactive accumulation is to calculate the best path to the current frame from the reference frame. For this we create a graph where each frame is a node and every pre-accumulation is a possible path, each path is given the same weighting so that length is the only factor in deciding which is the best route. We then use a greedy breadth first search to quickly determine the optimal route.

Once the optimal route has been calculated, all of the required frames must then be loaded into memory. This is necessary because we do not know which pixels will be accessed from each frame as we traverse back along the optimal path and so we cannot make use of Nuke's row-based image processing. Because we are using *pull* vectors we must begin at the current frame and work backwards until we reach the reference frame. These accumulations are calculated in our custom C++ plugin called VectorAccumulator. The MagicWarper tool is a Nuke gizmo which incorporates the VectorAccumulator plugin along with some other functionality to deal with the various options that a user can select.

The inputs to the MagicWarper tool are the sequence, the vectors, and the element to be added to the sequence. The user must simply position the element where they want it and specify the reference frame for which the element has been positioned. Once this is done it is simply a case of selecting the mode (track or stabilise) and then skipping to another frame to check the result. Some example outputs are given in the results section below.

6.4.5 Track mode

When "track" mode is selected and the frame difference is positive (i.e. the current frame is higher than the reference frame), the algorithm will use the backward vectors in order to plot a route for each pixel from the current frame back to the reference frame. When the current frame is earlier than the reference frame then the forward vectors are used for this purpose. This is necessary because we are using pull vectors and so the accumulated result at the current frame will tell us which pixels to grab from the reference frame. This method comes with the disadvantage that we must process the entire frame and cannot limit our search in the first instance due to the fact that we simply have no idea where the corresponding pixels will be in each frame. This is unavoidable, but mitigated somewhat by the fact that we can process each pixel independently and so optimise the speed of our algorithm through parallelism.

6.4.6 Stabilise mode

There are many examples of digital image stabilisation in the literature, using various techniques in order to achieve a global reduction in "jiggle" motion throughout a sequence [Chang et al., 2002; Erturk, 2003; Vella et al., 2002]. Rather than attempt to calculate what is "jiggle" and what is desired camera motion, we simply leave it to the artist to select the area they wish to stabilise and whether they would like it stabilised in the x or y axis (or both). Rotational stabilisation is not provided at present.

When "stabilise" mode is selected and the frame difference is positive, the algorithm will use forward vectors in order to calculate the route of each pixel within the given mask in the reference frame to the necessary position in the current frame. When the current frame is earlier than the reference frame then it will use the backward vectors for this purpose. The advantage with this mode is that all of the mask pixels are known in the reference frame and this is our starting point so we can limit our processing to just these pixels. The disadvantage with this mode is that we must process the entire masked area at once, since we then need to calculate the average motion values and apply those to the entire frame. This means that the size of the mask given will have a direct impact on processing time.

6.4.7 Reverse mode

This can be applied to either of the other two modes and will simply result in them performing the opposite action. This is particularly useful, and has been used in production, for face replacement. This is because we can use this process to warp a sequence (usually heavily masked out to just leave the face, or whatever else we are interested in) so that all of the pixels are where they should be in the reference frame, before further warping them using the regular track mode to add them to the new sequence. This gives the major advantage of being able to transfer animations along with the face replacement.

When used with stabilise mode, this function will simply accentuate any movements of the masked area and add them to the rest of each frame. While we are yet to use this in a real-world production, it could feasibly be used in a film where camera shake is desired but wasn't satisfactorily captured at the time.

6.4.8 Results

In order to demonstrate this tool we use test footage from the movie Spectre. We present every 50th frame from a 200 frame sequence where the only addition was made at frame 51, this demonstrates that this tool can work backwards just as effectively as forwards. Figure 6-8 shows all of the original images, while figure 6-9 shows the same images with an added element, which was painted on in the frame shown in the top right corner (frame 50 of the sequence).

We also show in figure 6-11 the limitation of our tool. It can only be as good as the vectors available, and when they become unreliable then the elements added to the scene can become unreliably warped. It is because of this that we have designed our tool to be able to take arbitrary keyframes throughout a sequence, so that when the vectors begin to drift and micro-deformations get more severe, then the artist can add in another keyframe and still save a lot of the manual work that would otherwise be necessary.

These results can be compared to those in figure 6-10 which show the output from The Foundry's VectorDistort node, which uses vectors created by their SmartVector node. It can be seen that the results are very similar in this case, with each displaying small (though differing) deformations at higher frame ranges. It should also be noted that while VectorDistort requires specific vectors to be created by the SmartVector node, MagicWarper can

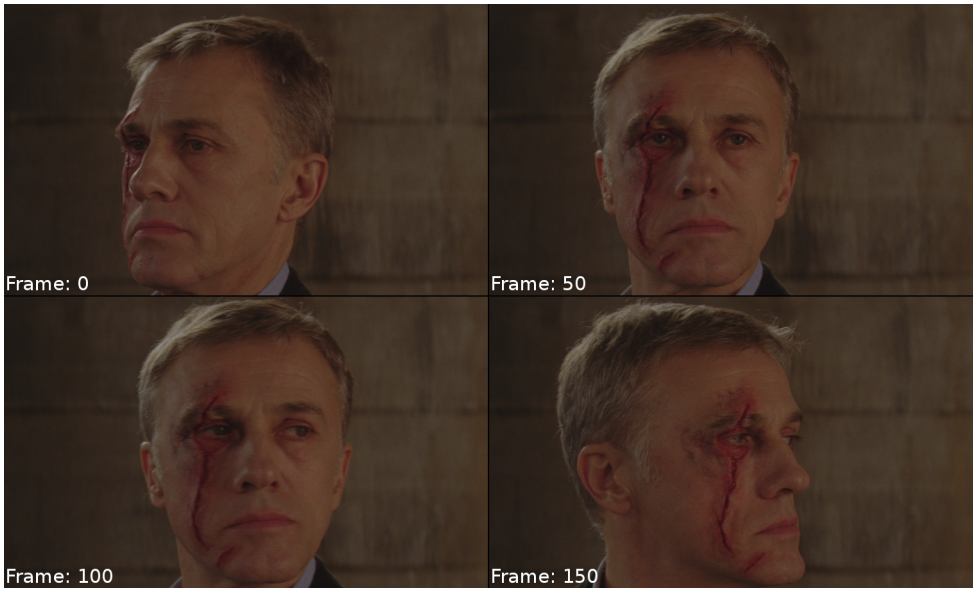

Figure 6-8: Four frames, taken at 50 frame intervals, showing the original images which we wish to add our chosen effect to.

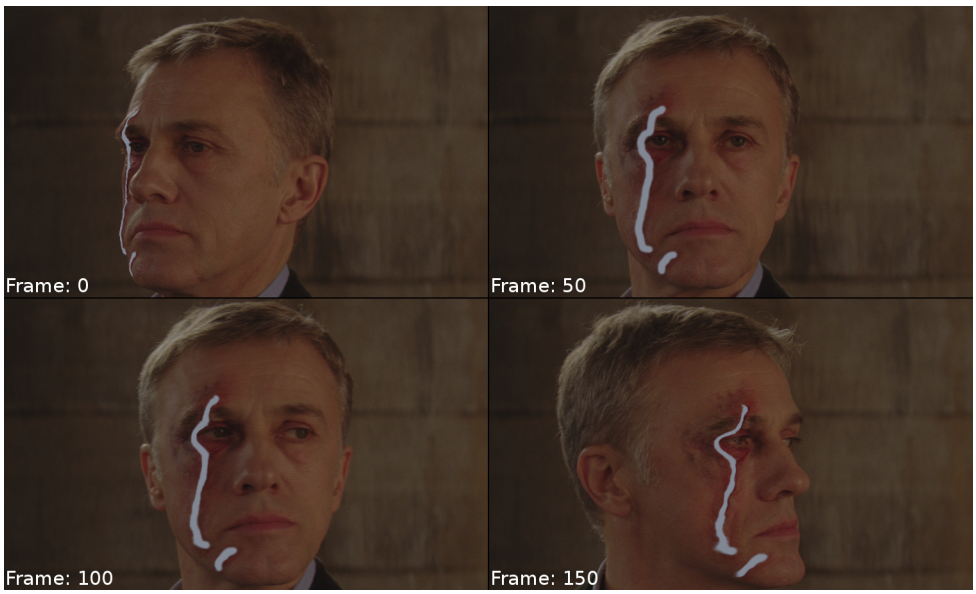

Figure 6-9: The same four frames as shown in figure 6-8. We simply painted over the scar in frame 50 and used the MagicWarper tool to propagate this to all of the other frames.

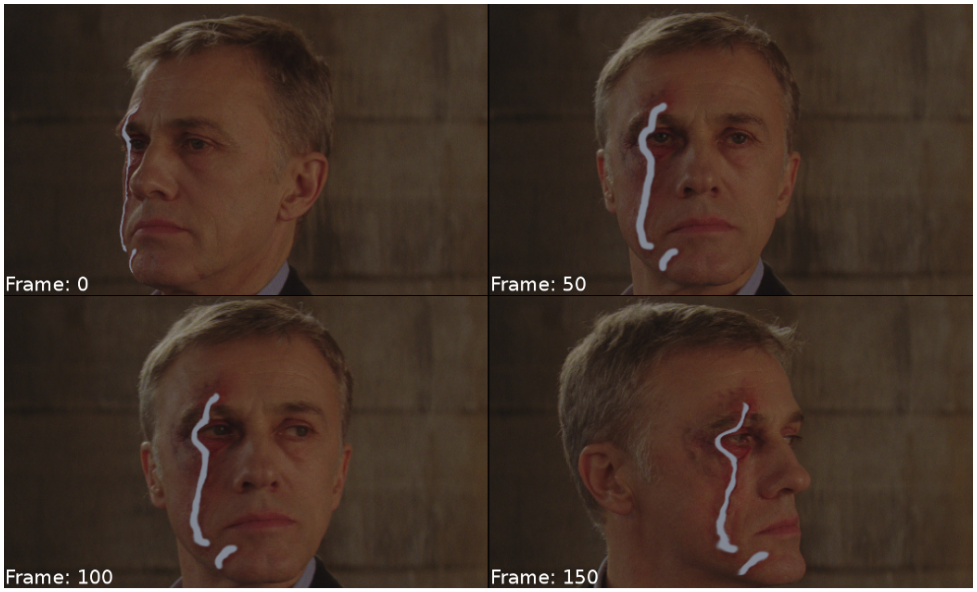

Figure 6-10: The same four frames as shown in figure 6-8. We simply painted over the scar in frame 50 and used the VectorDistort tool to propagate this to all of the other frames using vectors created by the SmartVector node.

take any appropriate vectors that will already be available for a sequence within the pipeline.

We demonstrate the stabilisation function by using the same Spectre sequence with the same element used this time as the region of interest from which to extract stabilisation vectors. We show in figure 6-12 three of the original frames, where the green crosshair has been placed at the exact same position. In figure 6-13 we show the same frames, with the crosshairs in the same position, after performing our stabilisation algorithm on the sequence (stabilising both x and y movement).

In figure 6-14 we show a production still from the TV series Agent Carter. For this scene we have a character who becomes enveloped in ice, an effect we are able to achieve with some creative use of the MagicWarper node. By creating a single frame of the ice effect, shown in figure 6-15, we are able to fill the entire sequence simply by setting this frame as the reference frame for MagicWarper. The merged image can be seen in figure 6-16, with a later frame shown in figure 6-17. It is also possible to animate the element for additional effects throughout the sequence. For example, a "reveal" effect can be achieved by applying a mask to the original frame that gradually shrinks through the framerange. For the Agent Carter sequence shown, a "sparkle" effect was added to the element sequence to add an extra layer to the look of the ice. Our tool allows the artist to forget about matching the

Figure 6-11: At 200 frames after the original paint was added, we begin to see errors occurring due to drift caused by accumulated errors in the vectors. At this point we must either add another keyframe, or mask out the erroneous areas and continue to use the rest.

Figure 6-12: Three frames from the same sequence as used to demonstrate the tracking mode (figure 6-8). The green crosshair is placed at the same x,y coordinate in each frame.

Figure 6-13: Following the use of the stabilisation mode of MagicWarper, it can be seen that the green crosshair (which remains in the same position as shown in figure 6-12) is now placed over the same area just beneath the actor's eye.

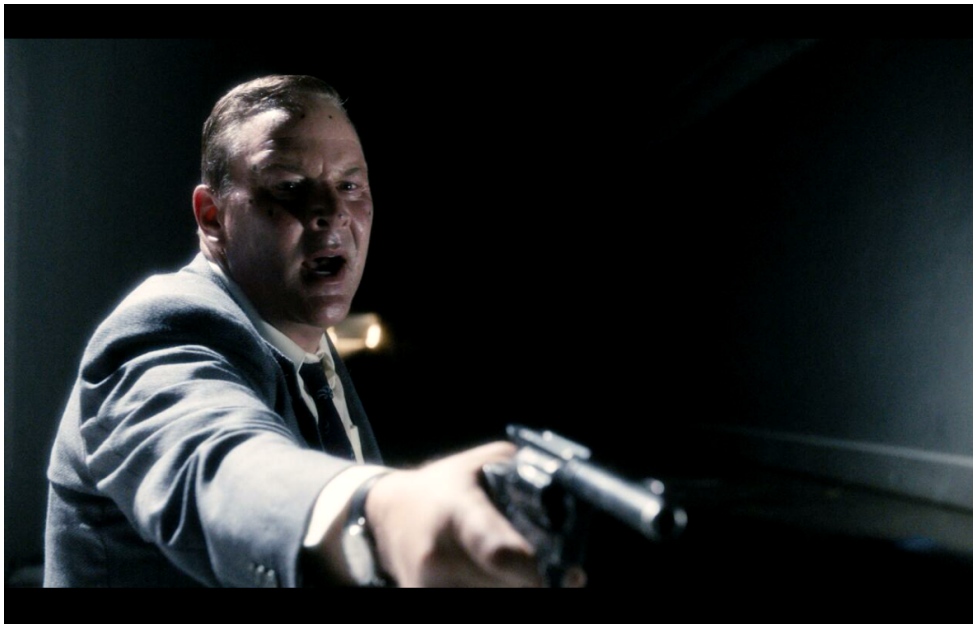

Figure 6-14: The original first frame of a sequence from Agent Carter.

movements of the element to those of the sequence and simply focus on the look of the element itself.

6.4.9 Discussion

We have shown the results produced by our MagicWarper tool, and it's 3 main functions. Of these, tracking has been the most rigorously road tested as it has been used on several major film and TV projects since mid-2015. The reverse and stabilise functions, however, have only been in use since mid-2016 and so are still going through the initial phase of adoption. That said, we have already received a lot of very positive feedback regarding both of these new features with uses being found in many of our projects. We compare our results to those produced by The Foundry's SmartVector tool, where appropriate. As far as we are aware, there is no publicly available tool that performs stabilisation in this way and so we are unable to compare these results. Similarly we are unaware of another tool that implements the "reverse" function that we have provided. While The Foundry's tools also allow for the addition of arbitrary keyframes, we have hidden more of the complexity of this and made it more intuitive for new users. Figure 6-18 shows the node graph required for any number of keyframes using the MagicWarper node, while figure 6-19 shows the necessary node graph for 2 keyframes. We achieve this simpler node graph by containing the necessary merge operations within the MagicWarper node, and simply using

Figure 6-15: A digital matte painting created for a single frame of the sequence.

Figure 6-16: The finished first frame, with the effect applied.

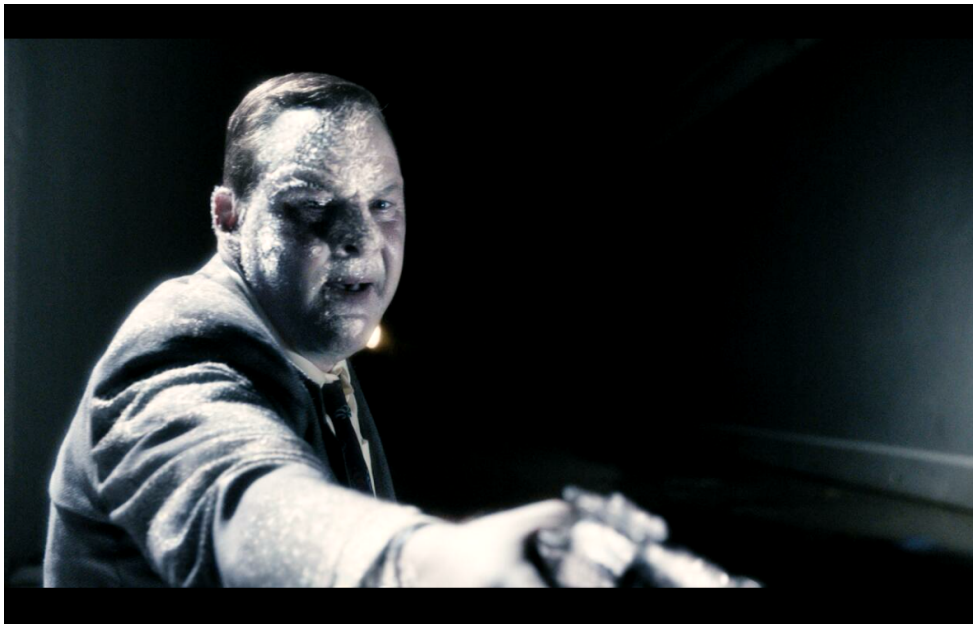

Figure 6-17: Another frame from further along in the sequence, with the motion of the digital matte painting calculated and merged onto the original image using MagicWarper.

the lifetime of the RotoPaint element along with the standard method of adding keyframes within a Nuke node.

6.4.10 Limitations

As previously mentioned, the accumulated motion vectors can only be as good as the original motion vectors supplied to our algorithm. It is for this reason that we have allowed it to take as input any arbitrary vectors so that several different methods of creation can be tried if necessary. There is really nothing that can be done regarding the quality of the input vectors apart from investigating our own methods of motion vector generation, but if we do that then our chosen technique will most certainly come with its own set of limitations that will prove restrictive in certain cases. For this reason we simply accept this as a limitation and leave it up to the artist to insert keyframes as appropriate to mitigate its effect.

Even with high quality vectors we often see some unintended micro-deformations in the given effect provided by the artist. An example of this can be seen in figures 6-9 and 6-10 where the effect appears smooth in early frames but becomes more jagged in later frames. This effect could be due to the original vectors provided, or it could be more to do with the sampling technique we use to accumulate them.

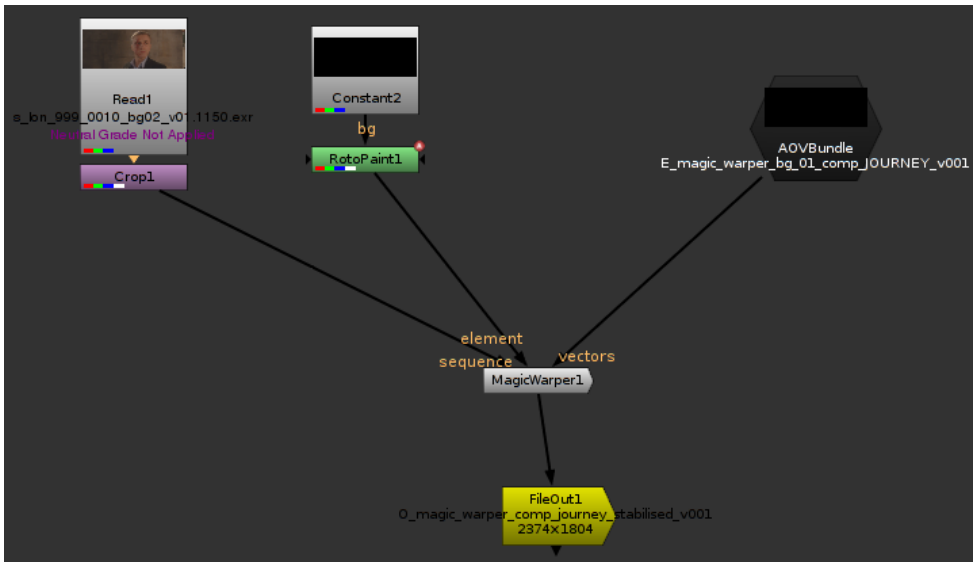

Figure 6-18: The Nuke node graph required for any number of keyframes using MagicWarper.

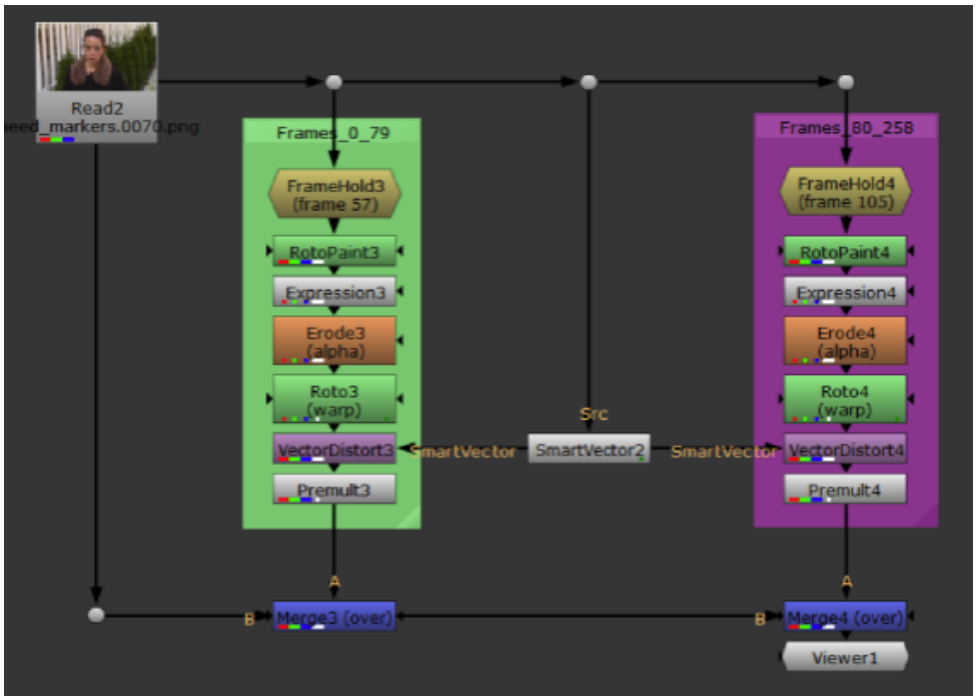

Figure 6-19: The node graph required for 2 keyframes using VectorDistort.

One area that could be improved is the sampling technique used when accumulating the motion vectors. At present we simply take the coordinates given by the motion vectors and take one pixel's worth of data surrounding that point, meaning we sample between +-0.5 of the given coordinates. There is potentially vast room for improvement in this technique which could reduce the deformations that we see appearing at frames far from the given reference point.

6.4.11 Conclusion

We have presented a novel tool for motion vector accumulation, as well as utilising those accumulated motion vectors for image warping and stabilisation. We have demonstrated the accuracy of our algorithm and shown many of the possible results from our MagicWarper tool as well as discuss its use in the movie post-production pipeline. In examining our method in detail we have highlighted a possible area for improvement in our vector sampling technique and hope to further explore this in the future.

By pre-accumulating the motion vectors in power-of-2 layers, we are able to maintain a more consistent speed for processing throughout the frame range of any sequence. By using a greedy breadth first search on a graph of the available accumulations we are able to quickly identify a suitable route to achieve the necessary result and make this tool usable in an interactive program. While we are far from able to provide real-time results, we are able to keep the maximum processing time per frame low enough to class this tool as interactive.

Having identified disk space usage as a major concern with our dense dataset approach, we then implemented a sparse version which would only take up a maximum of twice the amount of space of the original vectors. This resulted in a slight slow-down of our algorithm as the optimal paths became slightly longer in some cases, but due to the fact that we simultaneously implemented the ability to overshoot and backtrack when accumulating vectors (i.e. mix backwards and forwards vectors as necessary) we were also able to create shorter optimal paths in many other cases.

6.4.12 Future work

As previously mentioned, the latest version of this tool has only recently been released and so we are awaiting feedback from the artists to find out what further improvements are needed. In the meantime there are no major

plans for further development.

One possible improvement would be to avoid re-performing accumulations where they are not necessary. When rendering a sequence in interactive mode, we currently calculate the accumulations for each frame before creating the output and moving on to the next frame. It should be possible in this case to simply use the result of the previous frame and accumulate that with the current frame, meaning that after the first frame each following frame will only require one further calculation.

Additionally, we are investigating the possiblity of including a rotational stabilisation option in addition to the translational adjustment currently offered.

6.5 Other tools

There have been several other tools developed as smaller projects. These were largely completed towards the beginning of the EngD either as an exercise to improve programming, or as a means to become familiar with the pipeline and workflow at the company. These tools were created for a variety of software packages and have been used in some major projects. Most of this work was written in C++, along with some Python. While they do not contribute significantly towards any aspect of the major research projects described in this document, they were invaluable as stepping stones towards taking on and ultimately completing the larger projects. They also represent a significant contribution to the company in the form of previously non-existent time-saving tools that are still in use several years after creation.

6.5.1 ProjectionDeformer

This tool was first developed for Autodesk's Maya[(1)], which is used for 3D animation and modelling, and later converted to a Nuke node where it was found it could also be of use. ProjectionDeformer allows an artist to deform some geometry in such a way that it will continue to look exactly the same from a given perspective. This was designed to aid with stereo conversion on a show which had already been completed. The reason a tool like this is necessary is because a key part of converting single view images to stereo is to first recreate the geometry within the scene, and then project the original

[(1)] http://www.autodesk.co.uk/products/maya/overview

image onto it. From here it is possible to recapture the scene with a virtual camera moved slightly to one side. An artist will create the geometry in such a way that the original view looks exactly correct, but occasionally this can lead to under- or over-painting in some areas when the view is shifted. This tool allows the artist to make changes to the geometry as a means to remove these erroneous areas in the new view.

This node takes as input two cameras, the geometry to be deformed, and the projection plane for that geometry in order to create a scene, as shown in figure 6-20. There are several options available for the user to control the deforming of this geometry, these are shown in figure 6-24. The first camera is known as the "projection" camera, this is the position of the original camera for which we have real footage. The original footage will be projected on to the geometry from this angle. The second camera is known as the "render" camera and is the new view that we are trying to create. The geometry will always be deformed in such a way that it looks exactly the same from this angle.

Figure 6-24 shows the various options available to the user. The main controls are "envelope", "scale", and "offset". The first control, "envelope", fades the effect of the deformation such that 0 results in no deformation being performed at all. Secondly, "scale" determines the degree by which the geometry is stretched toward the render camera. Finally, "offset" will move the entire geometry closer or farther from the camera. Figure 6-21 shows the scene with an envelope, scale and offset of 0, while figure 6-22 shows the opposite effect. Regardless of these settings, the view from the render camera remains that shown in figure 6-23.

6.5.2 Line

This node was developed for Nuke and served as the perfect training exercise for future developments. It was written using C++ and making heavy use of OpenGL. It allows a user to define 2 points on an image and set various parameters to draw a line. While the node appears to be very simple, the use of OpenGL as well as additional features such as rounded ends, tapering, and blurred falloff, meant that it was a very good project for learning a lot of different aspects of development within Double Negative and specifically Nuke. Figure 6-25 shows a line being created by a user, the outline and points, as well as their labels, are produced with openGL. Figure 6-26 shows the same line as it will be rendered, without the outline which is only produced for guidance. Figure 6-27 shows the node panel in Nuke where users can set the parameters to the desired values. The coordinates of the start and end points can either be entered into the box

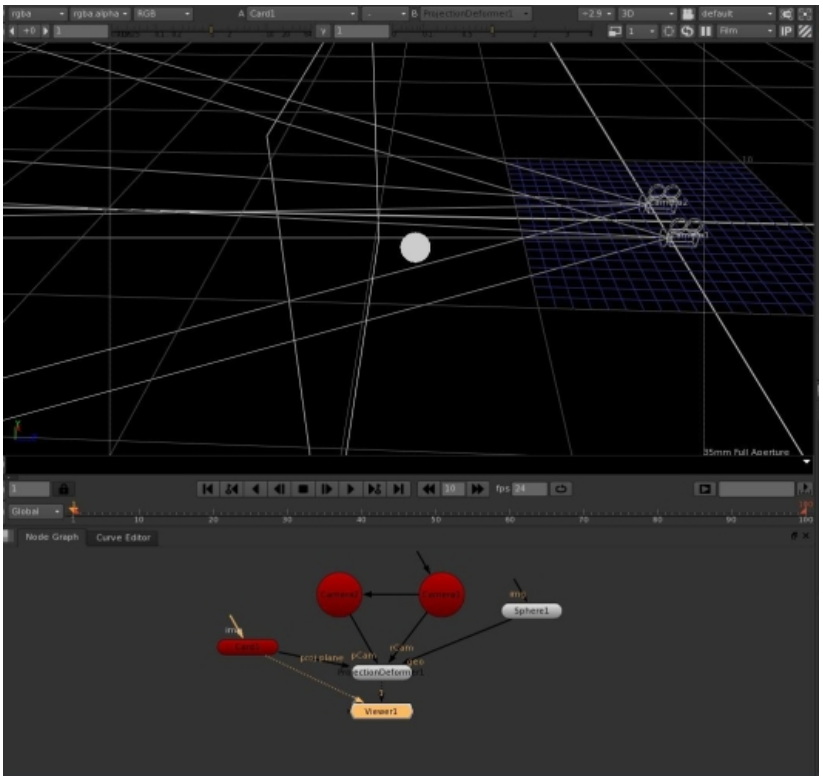

Figure 6-20: An example setup for the ProjectionDeformer node.

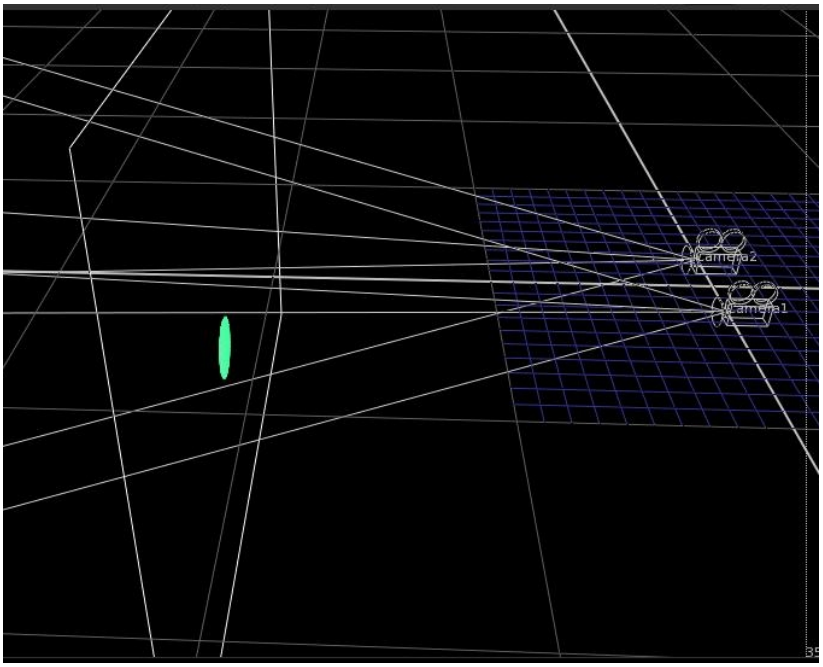

Figure 6-21: The example scene with an envelope, scale and offset of 0.

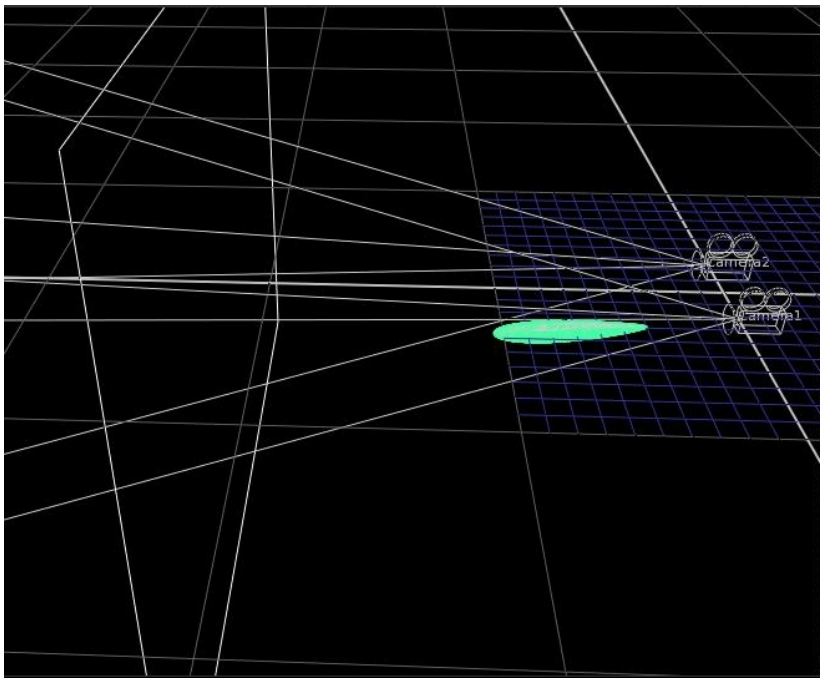

Figure 6-22: The example scene with envelope, scale and offset set extremely high.

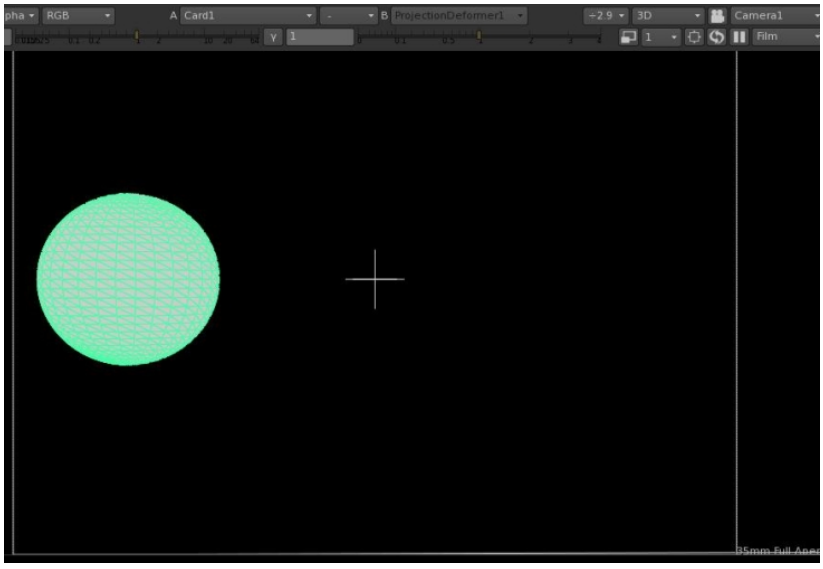

Figure 6-23: The view from the render camera, regardless of the values set for envelope, scale and offset.

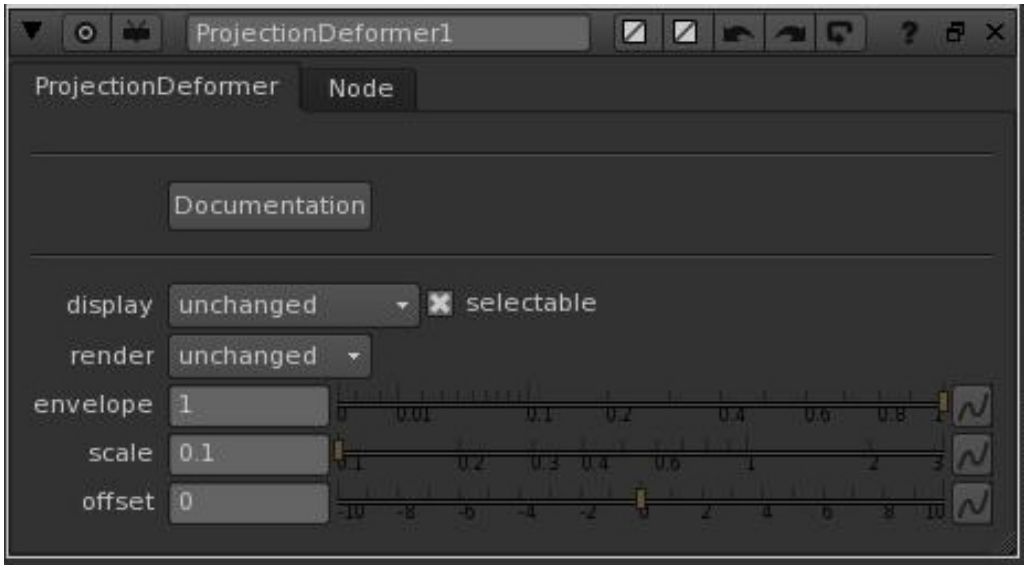

Figure 6-24: The UI for our ProjectionDeformer node.

or the OpenGL points can be dragged across the viewer. Start and end radius provide the thickness of each end of the line, while the nPoints value gives the roundness of the ends. The user can then specify a colour as well as the type of blur that they want, giving each end a different magnitude of blur if required. The "size" parameter refers to the maximum spread of the blur, and "mix" allows the artist to adjust the opacity of the line.

The line node has been used much more than initially anticipated. It has, in fact, been used on most Double Negative projects since it was created in early 2010. It has not required much in the way of maintenance for several years but recently (September 2015) requests have been made for some additional features to be added. These include the addition of arrow heads to be made possible so that this node could additionally be used as a diagnostic/feedback tool for shots in development.

6.5.3 Upgrading existing software to handle stereo images

This was a very effective way of learning about many tools used at Double Negative, as well as providing an important introduction to OpenEXR and the then-new SXR format for stereo image files. There is not much that can be said about this development as it was for existing tools which cannot be discussed, but a lot needed to be updated in order to accept and properly handle the new file format. This development required many changes being made to existing code and so proved to be a very good learning experience

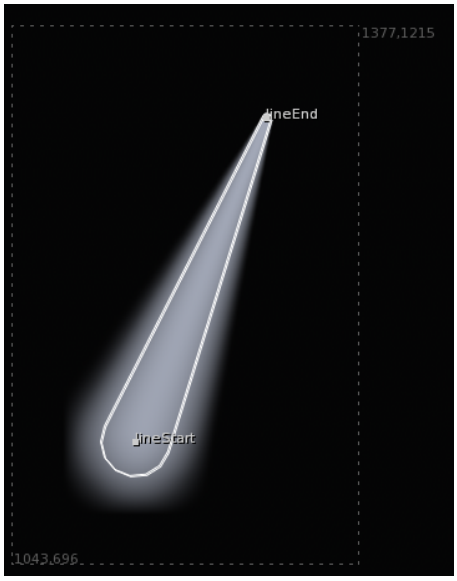

Figure 6-25: The line being created, with an openGL outline for guidance.

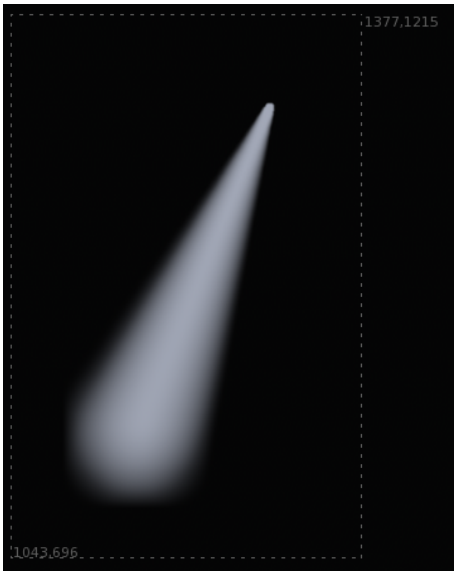

Figure 6-26: The line as it will be rendered.

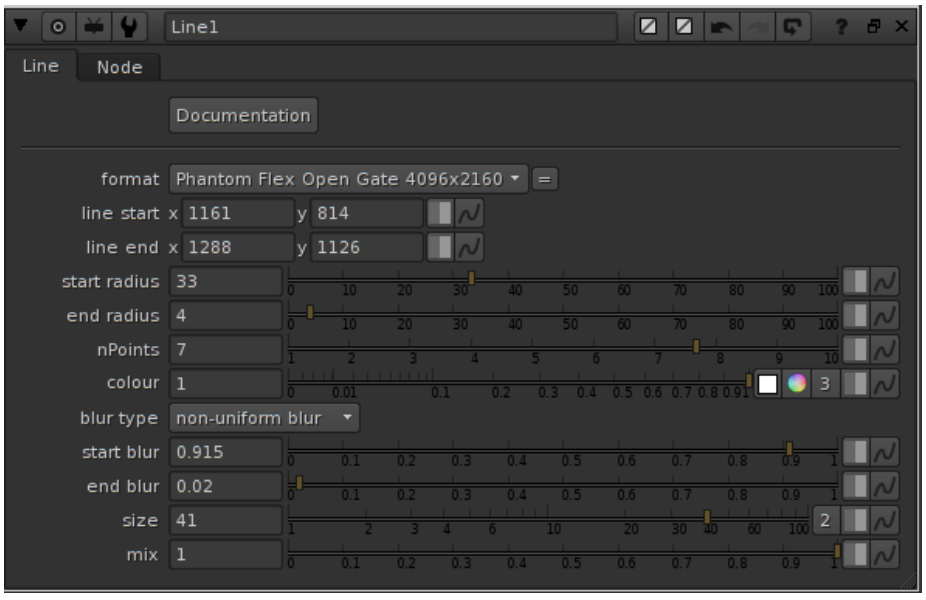

Figure 6-27: The parameters which can be edited by a user to create various effects.

and a great way to become familiar with many elements of Double Negative's workflow. It also allowed investigation into the OpenEXR format which proved invaluable throughout the research.

6.6 epiFlow: a work-in-progress standalone application for creating disparity maps between stereo images

This application is currently in development with the aim to produce better disparity maps for use by other tools, such as the EyeMatch and VerticalAlignment plugins discussed above. We define "better" as having sharper edges and less noise, while maintaining comparable accuracy to other disparity map generation techniques. We discuss the inner workings of this application, along with the alternative methods available in chapter 5. The decision was made to create this as a standalone application due to the large amount of image processing required. It will be possible to create this as a Nuke plugin in the future which will allow for interactive adjustment of parameters, but initially it will be acceptable to use a separate command and initiate batch jobs to create large sequences of disparity maps which can then be imported into Nuke.

6.6.1 Contribution of this work

Chapter 5 highlights the academic contribution of this work in the development of a new optical flow algorithm which utilises many of the constraints present in stereo filming to benefit the output. The biggest industrial contribution, however, is simply to provide other tools, such as EyeMatch and VerticalAlignment, with an improved input so that they can create a superior output. These are not the only tools at Double Negative that can benefit from disparity maps specifically made for stereo images, and there are many areas that will be able to produce better results as a consequence of its development.

6.6.2 Description of this program

The epiFlow tool takes as input simply the left and right view of a stereo pair of images. The full algorithm for creating the output is explained in

detail in section 5.4 with further plans to create an interactive disparity map generator provided in section 5.8. This program was created as a standalone application in order to allow experimentation during development without being concerned with restrictions within Nuke. This freedom during development has allowed us to make heavy use of the OpenCV library, which has in turn allowed us to save time on routine implementations and try various methods for achieving our desired outcomes. While this program is currently more of a testbed than a tool, further research is planned and development of a new disparity map generator will follow.

6.6.3 Future work

Development of this tool is ongoing, and so there are many elements that still need to be added and improved. Our chief aim is to produce an output that can be used in the post-production pipeline, and good progress is being made in this regard. This is a tool that Double Negative is very keen to develop as it has the potential to greatly improve many existing ones without disrupting their current way of working.

It is also possible that this tool could be of use in another new area of development at Double Negative – 360° camera rigs, and virtual reality. The use of 360° camera rigs for environment capture has been an area of interest for several years, but has recently received much more attention due to the increased capabilities of products such as the Oculus Rift[(2)]. There are many possible applications for this area of research, whether for a fully rendered immersive experience or for quick feedback to the director while onset so they can see how their characters will look in the CG environment. This sort of project would require all of the tools discussed in this dissertation, but they would need to be able to handle many more views – this is the next stage once we have a working disparity map generator.

There are many constraints that have been used with the knowledge that it is to be used on stereo images alone, but there are several occasions where it may in fact be possible to use it for a full sequence of mono images (and ultimately combine the two for a full sequence of stereo images). As further work for this tool, it would be good to see if we can relax some of the constraints in order to also use it from frame to frame. This would be another area that Double Negative would be keen to explore, since this would greatly broaden the impact of this work to all 2D films in addition to the 3D films for which it was developed.

[(2)] www.oculus.com

6.7 Industrial impact

In this chapter we have presented several tools that have been developed for use in the industry of visual effects production. These tools have all been incorporated into the pipeline at Double Negative and used on large scale movie projects. Where available, we have presented data which demonstrates the use of these various tools by individuals or projects and provided example outputs. The MagicWarper tool alone has been used on many different projects ranging from blockbuster films such as The Avengers 2 and Fantastic Beasts and Where to Find Them, to TV programmes such as Agent Carter and BrainDead. It can clearly be seen from the evidence provided in this chapter that the work presented has had a sizeable impact on the Double Negative pipeline and allowed complex tasks to be completed much quicker and easier than was previously possible.

Chapter 7

Conclusion

7.1 Discussion

The projects in this dissertation have covered several areas that present problems in the production of visual effects for films, and also highlighted issues within the existing pipeline at Double Negative. Working with a company that already has a very well established Research and Development team meant that there was a wealth of knowledge and experience (as well as prior work) that could be drawn on whenever necessary to further this research.

Before the first major research project was undertaken, there was a period of time spent learning about the Double Negative pipeline and the programming languages used there. During this time, several tools were created that proved to be of use to the company and have been used ever since on various projects. These tools are detailed in section 6.5 and represent approximately six months at the beginning of the research period. During this time there were many discussions regarding the long term direction of research, and the first major project was agreed upon.

The EyeMatch project (chapter 3 and section 6.2) was the first to be undertaken and was ongoing for approximately 18 months. This time includes all of the foundation research that was used for the other projects as well. When work began on the EyeMatch tool, there was no established way of working on stereo problems at Double Negative. This was because 3D films had only recently started to become popular again and many of the 3D projects at the time were still trying to "cheat" the effect by converting 2D footage post hoc. Apart from test footage, which was never used for a film, the EyeMatch tool was first successfully used for Aardman Animation's

"The Pirates! In an Adventure with Scientists!". It was also later used extensively for the Ridley Scott movie "Exodus: Gods and Kings", proving its worth within the Double Negative production pipeline and validating the industrial impact of the research. In addition to this, a paper entitled "Colour Matching Between Stereo Pairs of Images" was accepted into the User Centric Computer Vision portion of the Asian Conference on Computer Vision held in Singapore in 2014 [Willey et al., 2014]. The main focus of this sub-conference is the actual usability of computer vision research, and so it felt like the most appropriate outlet for research being conducted with a view to being used in industry for real-world projects. In addition to this, a further paper was produced for CVMP 2016 [Willey et al., 2016] covering the more technical aspects of the work.

Once the EyeMatch project was completed, it was decided that the next problem that was currently lacking a good solution was the vertical misalignment issue often present between stereo images (chapter 4 and section 6.3). Research of this problem offered a valuable learning experience in everything that is required to make a 3D image properly viewable, as well as the interesting differences between solutions required for end results and those required when the artist will be completing further work on the images. This project differed from the EyeMatch project in that several solutions already existed. While the Nuke suite of stereo plugins, Ocula, was not yet available, there were still methods employed by artists that would achieve good results. Ocula was actually released around the time of completion for this project, however we had implemented a technique that we believe to be valuable that The Foundry had not included in their solution. The "group by x-disparity" mode of our VerticalAlignment plugin has the capacity to create very good results and is still used by artists in preference to those solutions provided by The Foundry for some problems.

Finally, the epiFlow project was undertaken in order to bring everything together and create a complete suite of interdependent tools that could be used to perform a large part of the processing required for stereo movies. It was discovered during the colour matching and vertical alignment projects that the major limiting factor to each solution was the quality of the disparity map used. While the disparity maps produced by The Foundry's DisparityGenerator are adequate for many tasks and produce good results, we wanted to investigate further in order to see if there were any improvements that could be made to these. Specifically for the purposes of using them as inputs to our previous two major tools, and perhaps to other areas of the post-production pipeline. At the time that this project was started, it was understood that creating a fully polished implementation in the time remaining would be unlikely. However, significant research and development has been achieved towards creating a tool that could become a part of the current stereo pipeline.

We believe we have created some valuable work that has provided both an academic contribution and industry impact. There is a substantial amount of future work that could be completed in order to further the usefulness of the tools created, but they are already very useful in their current state. We have identified and discussed the future work that could be completed for each tool at the end of their respective chapters (3 - 5) and offer a more general assessment of future work that could be completed with a broader scope here.

7.2 Conclusion

This dissertation documents our investigations into whether the post-production pipeline for stereo movies can be improved through technical research in various areas. We have presented our work in the areas of colour matching, vertical alignment, and disparity map generation, as well as other projects, along with our conclusions and how best to continue this work for the further benefit of the industry and academia.

A summary of the contributions of this work is as follows:

- Local colour matching of high resolution stereo images (chapter 3)
  - Colour accuracy compares favourably to existing methods
  - Preserves noise
  - A Nuke plugin for seamless integration into the post-production pipeline
- Vertical alignment correction (chapter 4)
  - A collection of methods which allow for global and local correction
  - Grouping based on x-disparity similarity for y-disparity correction
  - A Nuke plugin which brings together several methods of alignment correction
- Disparity map generation (chapter 5)
  - Investigation of a new method to achieve cleaner edges and smoother disparities

- Warping with accumulated motion vectors (section 6.4)
  - A Nuke plugin for propagating elements throughout a sequence from minimal keyframes, or stabilising specified areas within a scene
- An in-depth exploration of the requirements of a 3D film

The colour matching problem was completely new for Double Negative when we started our investigation, and we are confident that our solution has been extremely beneficial to the stereo pipeline. This is evidenced not only in the fact that the tool was used for the movie "The Pirates! In an Adventure with Scientists!" (2012) but also through its use several years later for the movie "Exodus: Gods and Kings" (2014). These are the only two projects that Double Negative has worked on that were filmed with a stereo camera setup, but more will almost certainly be coming in the future. Both of these movies also made use of the VerticalAlignment tool, as it offers alternative functionality to other available tools.

One thing to note about this kind of work is that much of it is subjective, and so analysing results can sometimes be challenging. During development we were fortunate to have the cooperation of several artists who were willing to test our tools and provide feedback. This is the largest benefit to performing research in an industrial setting, unfortunately it can also be the biggest drawback. This is due to the fact that sometimes what the artist requires is not groundbreaking research with an original contribution, but something that will be available within a day to aid them in meeting an upcoming deadline. Finding the balance between these things has been a big challenge in itself.

The development of this dissertation has been fairly organic in that during work on the initial set of tools which formed the training and orientation part of the industrial placement, the need for colour matching between stereo pairs of images became apparent. This led to the formulation of the first major project, which is covered in chapter 3. While working on the colour matching problem, after it was decided that the disparity map route would be the favoured option, it was discovered that vertical misalignment was a common issue with our test images, and so the second project was decided on as a natural continuation of the first in that both projects are aimed at making viewing a 3D movie a better experience. Another issue that became apparent through the course of the first project, and later throughout the second project, was that of the disparity maps themselves. It was clear that over reliance on disparity maps meant that if these were lacking in quality in any way then the results of both projects would be severely affected. This led to the third project of investigating a new dis-

parity map generation algorithm to improve upon those already in use at Double Negative. During these investigations into disparity maps the idea of using motion vectors for propagating elements through a 2D sequence came up, and so the MagicWarper project was started. This organic way of working from project to project has worked well in this case. The rise of 3D movies created an opportunity to be at the forefront of an exciting new branch of VFX that had never been an issue for Double Negative before. And the fact that one problem naturally kept leading to others ultimately led to a rounded body of work which we have presented in this dissertation.

This dissertation makes a clear case that technical research is important for a company such as Double Negative. Visual Effects require a large amount of computing power and technical know-how even at the most basic level, and major "blockbusters" often require much more. New techniques and algorithms are constantly being developed by the Research and Development team in order to solve new problems, or to optimise old solutions. The projects discussed in this dissertation have provided a contribution to the company in the sense that they have all been used on major film projects and been useful to the artists in the process. They have provided the level of results needed, and in an appropriate time frame. In order to remain competitive, visual effects companies will always need to stay at the forefront of research in many areas in order to develop the new tools that are necessary to solve new problems that are just around the corner.

7.3 Future Work

In conducting this research, many areas have been touched on that would benefit from further exploration that is outside the scope of this dissertation.

We have noted the lack of studies into the discomfort caused by viewing erroneous 3D, and would like to see more. There have been several studies which have looked at 3D TVs, or virtual reality headsets, but none that have focussed on the cinema experience. Furthermore, the degree of error that can be tolerated in 3D images is something that would be of great interest. There have been studies which have looked at the effect of vertical misalignment, but we would be interested in finding a "comfort threshold" for colour mismatch between stereo images. We would like to know at what degree of colour mismatch viewing these images becomes uncomfortable, and whether there are perhaps certain colours which have a greater or lesser effect on the viewing experience.

Another area of interest would be that of 360° video which we briefly men-

tioned in chapter 5. We would be interested in seeing further research into the calculation of disparity maps to cover the full 360° set-up and the use of these in automatically aligning and stitching all of the views. Additionally, the colour matching problem from chapter 3 is only intensified with more cameras (current experiments at Double Negative use fourteen cameras, in a configuration of seven stereo pairs) and so would require further development. One additional problem with this is the potential lighting differences at various points on the circle, meaning that we cannot simply match all of the colours in all of the views but must develop a way of deciding when to adjust colours and when to accept changes. If these problems can be solved, then the production of live action environments for virtual reality will become a much more efficient process.

We were initially interested in exploring the use of the GPU in producing our disparity maps, and this continues to be on the development roadmap. However, we have also begun to explore the idea of using our research in disparity map generation software to inform future developments of hardware. Specifically, we are interested in calculating the necessary disparity information for panoramic stitching at the time of image capture.

Bibliography

Backus, B. T., Banks, M. S., van Ee, R., and Crowell, J. A. (1999). Horizontal and vertical disparity, eye position, and stereoscopic slant perception. *Vision research*, 39(6):1143–1170.

Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M. J., and Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31.

Banks, M. S., Read, J. C., Allison, R. S., and Watt, S. J. (2012). Stereoscopy and the human visual system. *SMPTE motion imaging journal*, 121(4):24–43.

Batlle, J., Mouaddib, E., and Salvi, J. (1998). Recent progress in coded structured light as a technique to solve the correspondence problem: a survey. *Pattern recognition*, 31(7):963–982.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer Vision–ECCV*, pages 404–417. Springer.

Beauchemin, S. S. and Barron, J. L. (1995). The computation of optical flow. *ACM Computing Surveys (CSUR)*, 27(3):433–466.

Bonneel, N., Sunkavalli, K., Paris, S., and Pfister, H. (2013). Example-based video color grading. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)*, 32(4):2.

Bouguet, J.-Y. (2001). Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, pages 1222–1239.

Brown, L. G. (1992). A survey of image registration techniques. *ACM computing surveys (CSUR)*, 24(4):325–376.

Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *Computer Vision–ECCV 2012*, pages 611–625. Springer.

Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698.

Chang, J.-Y., Hu, W.-F., Cheng, M.-H., and Chang, B.-S. (2002). Digital image translational and rotational motion stabilization using optical flow technique. *IEEE Transactions on Consumer Electronics*, 48(1):108–115.

Chen, C.-S., Hung, Y.-P., Chiang, C.-C., and Wu, J.-L. (1997). Range data acquisition using color structured lighting and stereo vision. *Image and Vision Computing*, 15(6):445–456.

Chen, Q. and Koltun, V. (2016). Full flow: Optical flow estimation by global optimization over regular grids. *arXiv preprint arXiv:1604.03513*.

Corrigan, D., Pitié, F., Morris, V., Rankin, A., Linnane, M., Kearney, G., Gorzel, M., O'Dea, M., Lee, C., and Kokaram, A. (2010). A video database for the development of stereo-3d post-production algorithms. In *Visual Media Production (CVMP), 2010 Conference on*, pages 64–73. IEEE.

Crum, W., Hartkens, T., and Hill, D. (2004). Non-rigid image registration: theory and practice. *British journal of radiology*, 77(Special Issue 2):S140.

Doshi, A. and Bors, A. G. (2010). Robust processing of optical flow of fluids. *IEEE Transactions on Image Processing*, 19(9):2332–2344.

Dosovitskiy, A., Fischery, P., Ilg, E., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T., et al. (2015). Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766. IEEE.

Elias, R. (2007). Sparse view stereo matching. *Pattern recognition letters*, 28(13):1667–1678.

Emoto, M., Niida, T., and Okano, F. (2005). Repeated vergence adaptation causes the decline of visual functions in watching stereoscopic television. *Display Technology, Journal of*, 1(2):328–340.

Erturk, S. (2003). Digital image stabilization with sub-image phase correlation based global motion estimation. *IEEE Transactions on Consumer Electronics*, 49(4):1320–1325.

Faridul, S. F., Pouli, T., Chamaret, C., Stauder, J., Trémeau, A., Reinhard, E., et al. (2014). A survey of color mapping and its applications. In *Eurographics (State of the Art Reports)*, pages 43–67.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fleet, D. and Weiss, Y. (2006). Optical flow estimation. In *Handbook of Mathematical Models in Computer Vision*, pages 237–257. Springer.

Forsyth, D. A. and Ponce, J. (2002). *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.

Freedman, B., Shpunt, A., Machline, M., and Arieli, Y. (2012). Depth mapping using projected patterns. US Patent 8,150,142.

Garg, R., Roussos, A., and Agapito, L. (2013). A variational approach to video registration with subspace constraints. *International journal of computer vision*, 104(3):286–314.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE.

Grundland, M. and Dodgson, N. A. (2005). Color histogram specification by histogram warping. In *Electronic Imaging 2005*, pages 610–621. International Society for Optics and Photonics.

HaCohen, Y., Shechtman, E., Goldman, D. B., and Lischinski, D. (2013). Optimizing color consistency in photo collections. *ACM Transactions on Graphics (TOG)*, 32(4):38.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Hasan, S. F., Stauder, J., and Trémeau, A. (2011). Robust color correction for stereo. In *Visual Media Production (CVMP), 2011 Conference for*, pages 101–108. IEEE.

Hoffman, D. M., Girshick, A. R., Akeley, K., and Banks, M. S. (2008). Vergence–accommodation conflicts hinder visual performance and cause visual fatigue. *Journal of vision*, 8(3):33.

Hooge, I. T. C. and Van den Berg, A. (2000). Visually evoked cyclovergence and extended listing's law. *Journal of neurophysiology*, 83(5):2757–2775.

Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics.

Horn, E. and Kiryati, N. (1999). Toward optimal structured light patterns. *Image and Vision Computing*, 17(2):87–97.

Ideses, I. and Yaroslavsky, L. (2005). Three methods that improve the visual quality of colour anaglyphs. *Journal of Optics A: Pure and Applied Optics*, 7(12):755.

Istead, L., Asente, P., and Kaplan, C. S. (2016). Layer-based disparity adjustment in stereoscopic 3d media. In *Proceedings of the 13th European Conference on Visual Media Production (CVMP 2016)*, page 2. ACM.

Khoshelham, K. (2011). Accuracy analysis of kinect depth data. *ISPRS workshop laser scanning*, 38(5):W12.

Kolmogorov, V. and Zabih, R. (2006). Graph cut algorithms for binocular stereo with occlusions. In *Handbook of Mathematical Models in Computer Vision*, pages 423–437. Springer.

Kooi, F. L. and Toet, A. (2004). Visual comfort of binocular and 3d displays. *Displays*, 25(2):99–108.

Kwatra, V., Schodl, A., Essa, I., Turk, G., and Bobick, A. (2003). Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286.

Lambooij, M., IJsselsteijn, W., and Heynderickx, I. (2011). Visual discomfort of 3d tv: Assessment methods and modeling. *Displays*, 32(4):209–218.

Lang, M., Wang, O., Aydin, T., Smolic, A., and Gross, M. (2012). Practical temporal consistency for image-based graphics applications. *ACM Transactions on Graphics (ToG)*, 31(4):34.

Longuet-Higgins, H. C. (1987). A computer algorithm for reconstructing a scene from two projections. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pages 61–62.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *The proceedings of the seventh IEEE international conference on Computer Vision*, volume 2, pages 1150–1157. IEEE.

Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679.

Luong, Q.-T. (1992). *Matrice fondamentale et calibration visuelle sur l'environnement. Vers une plus grande autonomie des système robotiques.* PhD thesis, Université Paris Sud-Paris XI.

Luong, Q.-T. and Faugeras, O. D. (1996). The fundamental matrix: Theory, algorithms, and stability analysis. *International journal of computer vision*, 17(1):43–75.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 416–423. IEEE.

Moisan, L., Moulon, P., and Monasse, P. (2016). Fundamental matrix of a stereo pair, with a contrario elimination of outliers. *Image Processing On Line*, 6:89–113.

Neundorf, J., Schmidt, V., Lagemann, T., and Hofmeyer, F. (2012). Automatic color matching between stereo images. In *Consumer Electronics-Berlin (ICCE-Berlin), 2012 IEEE International Conference on*, pages 185–189. IEEE.

Nielsen, K. and Poggio, T. (1984). Vertical image registration in stereopsis. *Vision Research*, 24(10):1133–1140.

Ochs, P., Malik, J., and Brox, T. (2014). Segmentation of moving objects by long term video analysis. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1187–1200.

Ogle, K. N. (1952). Disparity limits of stereopsis. *AMA archives of ophthalmology*, 48(1):50–60.

Ogle, K. N. (1955). Stereopsis and vertical disparity. *AMA archives of ophthalmology*, 53(4):495–504.

Okun, J. A. and Zwerman, S. (2010). *The VES handbook of visual effects: industry standard VFX practices and procedures*. Taylor & Francis US.

Pitié, F. and Kokaram, A. (2007). The linear monge-kantorovitch linear colour mapping for example-based colour transfer. In *Visual Media Production, 2007. IETCVMP. 4th European Conference on*, pages 1–9. IET.

Pitié, F., Kokaram, A. C., and Dahyot, R. (2007). Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding*, 107(1):123–137.

Pouli, T. and Reinhard, E. (2010). Progressive histogram reshaping for creative color transfer and tone reproduction. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 81–90. ACM.

Prazdny, K. (1985). Vertical disparity tolerance in random-dot stereograms. *Bulletin of the Psychonomic Society*.

Ray, S. and Turi, R. H. (1999). Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*, pages 137–143.

Reinhard, E., Adhikhmin, M., Gooch, B., and Shirley, P. (2001). Color transfer between images. *Computer Graphics and Applications, IEEE*, 21(5):34–41.

Ren, X. and Malik, J. (2003). Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 10–17. IEEE.

Revaud, J., Weinzaepfel, P., Harchaoui, Z., and Schmid, C. (2015). Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172.

Salvi, J., Fernandez, S., Pribanic, T., and Llado, X. (2010). A state of the art in structured light patterns for surface profilometry. *Pattern recognition*, 43(8):2666–2680.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., and Westling, P. (2014). High-resolution stereo datasets with subpixel-accurate ground truth. In *Pattern Recognition*, pages 31–42. Springer.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42.

Scharstein, D. and Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195. IEEE.

Senanayake, C. and Alexander, D. C. (2007). Colour transfer by feature based histogram registration. In *BMVC*, pages 1–10.

Sevilla-Lara, L., Sun, D., Jampani, V., and Black, M. J. (2016). Optical flow with semantic segmentation and localized layers. *arXiv preprint arXiv:1603.03911*.

Sevilla-Lara, L., Sun, D., Learned-Miller, E. G., and Black, M. J. (2014). Optical flow estimation with channel constancy. In *European Conference on Computer Vision*, pages 423–438. Springer.

Sezgin, M. et al. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–168.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905.

Shibata, T., Kim, J., Hoffman, D. M., and Banks, M. S. (2011). The zone of comfort: Predicting visual discomfort with stereo displays. *Journal of Vision*, 11(8):11.

Solimini, A. G., Mannocci, A., Di Thiene, D., and La Torre, G. (2012). A survey of visually induced symptoms and associated factors in spectators of three dimensional stereoscopic movies. *BMC public health*, 12(1):779.

Stevenson, S. B. and Schor, C. M. (1997). Human stereo matching is not restricted to epipolar lines. *Vision Research*, 37(19):2717–2723.

Sun, D., Liu, C., and Pfister, H. (2014a). Local layering for joint motion estimation and occlusion detection. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1098–1105. IEEE.

Sun, D., Roth, S., and Black, M. J. (2010). Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE.

Sun, D., Roth, S., and Black, M. J. (2014b). A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137.

Sun, D., Sudderth, E. B., and Black, M. J. (2012). Layered segmentation and optical flow estimation over time. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1768–1775. IEEE.

Sun, D., Wulff, J., Sudderth, E. B., Pfister, H., and Black, M. J. (2013). A fully-connected layered model of foreground and background flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2451–2458.

Tam, W. J., Speranza, F., Yano, S., Shimono, K., and Ono, H. (2011). Stereoscopic 3d-tv: visual comfort. *IEEE Transactions on Broadcasting*, 57(2):335–346.

Ukai, K. and Howarth, P. A. (2008). Visual fatigue caused by viewing stereoscopic motion images: Background, theories, and observations. *Displays*, 29(2):106–116.

Vella, F., Castorina, A., Mancuso, M., and Messina, G. (2002). Digital image stabilization by adaptive block motion vectors filtering. *IEEE Transactions on Consumer Electronics*, 48(3):796–801.

Wertheimer, M. (1938). *Gestalt theory*. Hayes Barton Press.

Wheatstone, C. (1838). Contributions to the physiology of vision–part the first. on some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London*, pages 371–394.

Willey, S., Willis, P., Clifford, J., and Waine, T. (2014). Colour matching between stereo pairs of images. In *Computer Vision-ACCV 2014 Workshops*, pages 289–298. Springer.

Willey, S., Willis, P., Waine, T., and Hall, P. (2016). Localised colour matching between stereo pairs of images. In *Proceedings of the 13th European Conference on Visual Media Production (CVMP 2016)*, page 10. ACM.

Yang, S.-n., Schlieski, T., Selmins, B., Cooper, S. C., Doherty, R. A., Corriveau, P. J., and Sheedy, J. E. (2012). Stereoscopic viewing and reported perceived immersion and symptoms. *Optometry & Vision Science*, 89(7):1068–1080.

Yano, S., Emoto, M., and Mitsuhashi, T. (2004). Two factors in visual fatigue caused by stereoscopic hdtv images. *Displays*, 25(4):141–150.

Yano, S., Ide, S., Mitsuhashi, T., and Thwaites, H. (2002). A study of visual fatigue and visual comfort for 3d hdtv/hdtv images. *Displays*, 23(4):191–201.

Zhang, H., Fritts, J. E., and Goldman, S. A. (2008). Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding*, 110(2):260–280.

Zhang, Z. (1998). Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision*, 27(2):161–195.

Zitova, B. and Flusser, J. (2003). Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000.

Appendices

Appendix A

Calculatinng epipolar geometry

Given two views of the same scene, we can calculate the epipolar geometry in order to represent the relationship between pixels in each image. To briefly explain what is meant by this, we borrow figure A-1 from Forsyth and Ponce [2002]. This diagram shows the crucial relationship between all of the elements of a stereo pair of images. For our purposes, it is the concept of the epipolar line which is the most valuable. For a point in one image of the stereo pair, the corresponding point lies somewhere along the epipolar line in the other image. Figure A-2 (again, borrowed from Forsyth and Ponce [2002]) shows this relationship where, regardless of whether p represents real-world point P, $P1$, or $P2$, the corresponding point always lies on the epipolar line l' in the other view.

Since we are unable to calibrate the cameras that are used on-set, and know nothing of their internal parameters, we must use the fundamental matrix to represent the camera motion between views and relate corresponding points between the stereo pair of images. The relationship between the fundamental matrix and the epipolar line is shown in equation (A.1).

$$Fx = l' \tag{A.1}$$

Where x is a point in image 1, l' is the corresponding epipolar line in image 2, and F is the fundamental matrix.

The *fundamental matrix* itself was first introduced by [Luong, 1992] as a generalisation of the *essential matrix* [Longuet-Higgins, 1987] which can only be used for calibrated cameras. The relationship between the fundamental matrix and the essential matrix is shown in equation (A.2).

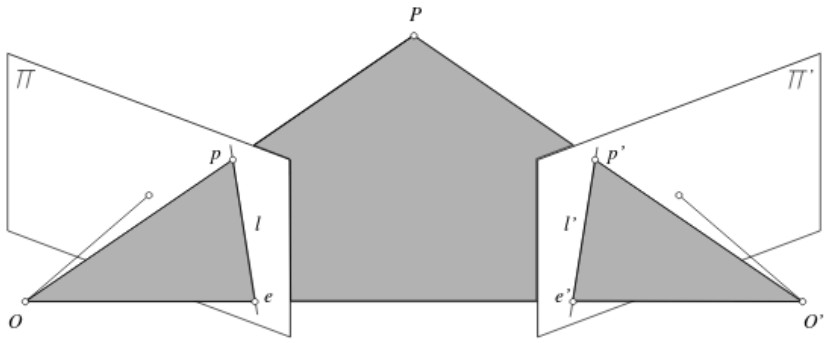

Figure A-1: A simple diagram to demonstrate epipolar geometry. Where O and O' are the optical centres of the two cameras, P is a point in 3D space, p and p' are the 2D representations of point P in each image, and e and e' are the epipoles (i.e. the point where the *other* camera's optical centre appears in each view). All of these points lie on the same plane.

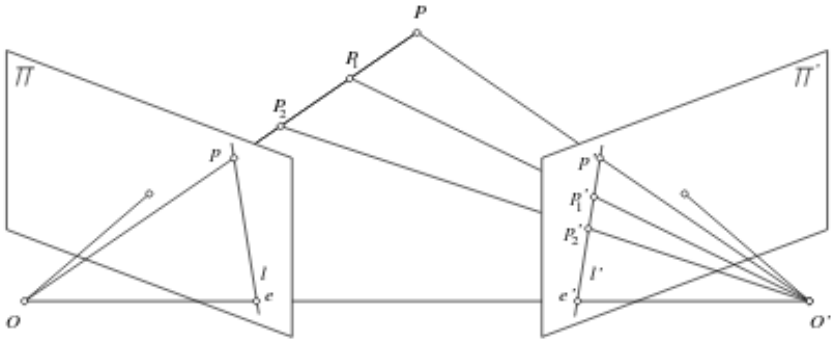

Figure A-2: A simple diagram to demonstrate the concept of the epipolar line. Real-world points P, $P1$ and $P2$ all lie on epipolar line l'.

$$E = K'^T F K \tag{A.2}$$

Where E is the essential matrix, and K and K' are the intrinsic calibration matrices for the two cameras.

We can take advantage of the fact that the corresponding point x' will lie on the epipolar line l' to rewrite equation (A.1) as:

$$x'^T F x = 0 \tag{A.3}$$

Which can in turn be written as:

$$U^T f = 0 \tag{A.4}$$

$$\text{Where: } U = \left[uu', vu', u', uv', vv', v', u, v, 1\right] \tag{A.5}$$

$$f = \left[F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}\right] \tag{A.6}$$

From equation (A.4) we can determine that 8 corresponding points will be sufficient to allow us to calculate the fundamental matrix. To do this we use the eight-point algorithm first put forward by Longuet-Higgins [1987] and developed for use with uncalibrated cameras by Luong and Faugeras [1996], which is shown in equation A.7.

$$\begin{bmatrix} u_1u'_1 & v_1u'_1 & u'_1 & u_1v'_1 & v_1v'_1 & v'_1 & u_1 & v_1 \\ u_2u'_2 & v_2u'_2 & u'_2 & u_2v'_2 & v_2v'_2 & v'_2 & u_2 & v_2 \\ u_3u'_3 & v_3u'_3 & u'_3 & u_3v'_3 & v_3v'_3 & v'_3 & u_3 & v_3 \\ u_4u'_4 & v_4u'_4 & u'_4 & u_4v'_4 & v_4v'_4 & v'_4 & u_4 & v_4 \\ u_5u'_5 & v_5u'_5 & u'_5 & u_5v'_5 & v_5v'_5 & v'_5 & u_5 & v_5 \\ u_6u'_6 & v_6u'_6 & u'_6 & u_6v'_6 & v_6v'_6 & v'_6 & u_6 & v_6 \\ u_7u'_7 & v_7u'_7 & u'_7 & u_7v'_7 & v_7v'_7 & v'_7 & u_7 & v_7 \\ u_8u'_8 & v_8u'_8 & u'_8 & u_8v'_8 & v_8v'_8 & v'_8 & u_8 & v_8 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \end{bmatrix} = - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{A.7}$$

Since in practice we will likely have more than 8 points, the least squares method is used to find the optimal solution.